Aumentando a produtividade com Android Libs

+Nelson Glauber
@nglauber
www.nglauber.com.br

C.E.S.A.R

FACULDADE UNIBRATEC

Intel Software Innovator

Google Developers Experts

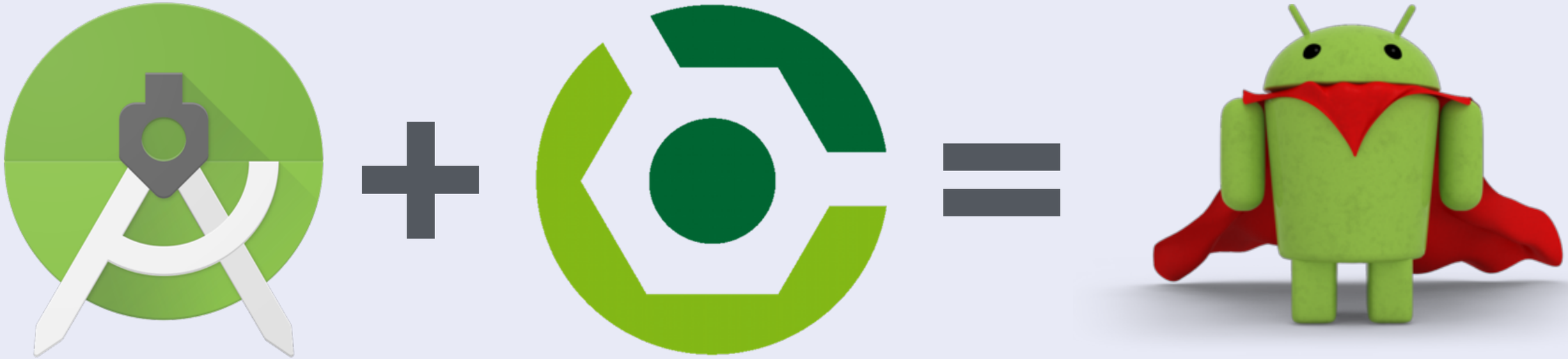
@nglauber

+NelsonGlauber

www.nglauber.com.br

O que vamos falar aqui...

- Apresentar as principais bibliotecas utilizadas no desenvolvimento de aplicativos Android.
- Experiência no meu projeto atual que usa (praticamente) todas!!! :O
- Disseminar esse conhecimento para acelerar o desenvolvimento de aplicações Android.



Esqueça...
Já passou...

#nostalgia

Support Libraries: appcompat

- Garantia de compatibilidade entre versões do Android.
- Possui os componentes das support libraries **v4** e **v7**
- Diversos componentes importantes: **AppCompatActivity**, **Fragment**, **Toolbar**, **NotificationCompat**, **ViewPager**, **DrawerLayout**, **SlidingPanelLayout**, **Loader**, **LocalBroadcastManager, ShareActionProvider**, ...

```
dependencies {
    ...
    compile 'com.android.support:appcompat-v7:22.2.0'
}
```

Support Libraries++

Os componentes **CardView**, **GridLayout**, **Pallete** e **RecyclerView** também estão disponíveis em bibliotecas de suporte!

```
dependencies {
    ...
    compile 'com.android.support:cardview-v7:22.2.0'
    compile 'com.android.support:gridlayout-v7:22.2.0'
    compile 'com.android.support:palette-v7:22.2.0'
    compile 'com.android.support:recyclerview-v7:22.2.0'
}
```

Butter Knife

http://jakewharton.github.io/butterknife/

```
dependencies {
    ...
    compile 'com.jakewharton:butterknife:6.1.0'
}
```

```java
@InjectView(R.id.edtFaca)
EditText edtFaca;

@InjectView(R.id.txtManteiga)
TextView txtManteiga;
```

```java
// Na Activity
public void onCreate(Bundle state){
    ...
    ButterKnife.inject(this);
}
// No Fragment
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_main, container, false);
    ButterKnife.inject(this, view);
    ...
}
```

```java
@OnClick(R.id.submit)
public void submit(View view) {

}
```

```java
@OnItemSelected(R.id.list_view)
void onItemSelected(int position) {

}
```

```java
TextView txtNome = ButterKnife.findById(view, R.id.txtNome);
```

```java
// Para fragments
public void onDestroyView(){
    ButterKnife.reset(this);
}
```

Data Binding

- Mapeia propriedades de um objeto no arquivo de layout.
- Lançada no Google I/O 2015.
- Funciona para Activities, Fragments e Adapters.
- Compatível com versões anteriores.
- Versão **BETA**!
- Mais detalhes: https://d.android.com/tools/data-binding/guide.html

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:1.3.0-beta1'
        classpath "com.android.databinding:dataBinder:1.0-rc0"
    }
}
allprojects {
    repositories {
        jcenter()
    }
}
```

```
apply plugin: 'com.android.databinding'
```

```
<layout xmlns:android="http://schemas.android.com/apk/res/android">
    <data>
        <variable
            name="livro"
            type="br.com.nglauber.intelsoftwaredaydemo.model.Livro" />
    </data>

    <RelativeLayout ...>
        <TextView android:text="@{livro.titulo}" ... />
        <TextView android:text="@{livro.autor}" ... />
        <TextView android:text="@{String.valueOf(livro.ano)}" .../>
        <TextView android:text="@{String.valueOf(livro.paginas)}" .../>
    </RelativeLayout>
</layout>
```

Acesso a web

http://square.github.io/okhttp/

https://code.google.com/p/google-gson/

```
dependencies {
    ...
    compile 'com.squareup.okhttp:okhttp:2.2.0'
    compile 'com.google.code.gson:gson:2.3.1'
}
```

```
OkHttpClient client = new OkHttpClient();

Request request = new Request.Builder()
        .url("http://seuservidor.com/livros.json")
        .build();

Response response = client.newCall(request).execute();
String json = response.body().string();

Gson gson = new Gson();
editora = gson.fromJson(json, Editora.class);
```

```
MediaType JSON = MediaType.parse("application/json; charset=utf-8");

OkHttpClient httpClient = new OkHttpClient();

Gson gson = new Gson();

String jsonReq = gson.toJson(livro);

RequestBody body = RequestBody.create(JSON, jsonReq);
Request postRequest = new Request.Builder()
        .url("http://seuservidor.com/post_livro")
        .post(body)
        .build();

Response postResponse = httpClient.newCall(postRequest).execute();
String jsonResposta = postResponse.body().string();
```

Acesso a web services REST++

Retrofit

A type-safe REST client for Android and Java

http://square.github.io/retrofit/

```
dependencies {
    ...
    compile 'com.squareup.retrofit:retrofit:1.9.0'
}
```

```java
import retrofit.Callback;
import retrofit.http.*;

public interface ProdutoService {
    String PATH_PRODUTOS = "/produtos";

    @GET(PATH_PRODUTOS)
    void listarProdutos(Callback<List<Produto>> callback);

    @POST(PATH_PRODUTOS)
    void adicionarProduto(@Body Produto produto,
        Callback<Produto> callback);

    @PUT(PATH_PRODUTOS+"/{id}")
    void atualizarProduto(@Path("id") int id,
        @Body Produto produto, Callback<Produto> callback);
}
```

```
RestAdapter restAdapter = new RestAdapter.Builder()
        .setEndpoint("http://meuservico.com/produtos")
        .build();
ProdutoService produtoService =
     restAdapter.create(ProdutoService.class);
```

```
produtoService.listarProdutos(
  new Callback<List<Produto>>() {
    @Override
    public void success(List<Produto> produtos,
                        Response response) {
        // É só ler a lista de produtos normalmente...
    }

    @Override
    public void failure(RetrofitError error) {
        // Erro ao obter lista de produtos
    }
});
```

```
Produto produto = new Produto();
produto.setDescricao("Computador");
produto.setPreco(12999.9);

produtoService.adicionarProduto(produto,
  new Callback<Produto>() {
    @Override
    public void success(Produto produto, Response response) {
        // Produto inserido com sucesso
    }

    @Override
    public void failure(RetrofitError error) {
        // Falha ao inserir produto
    }
});
```

Carregamento de imagens

Picasso

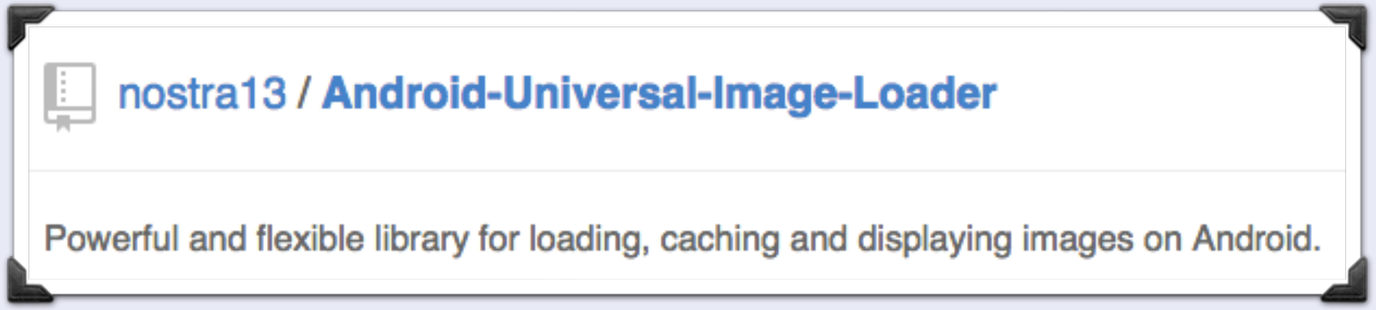
A powerful **image downloading** and **caching** library for Android

http://square.github.io/picasso/

```
dependencies {
    ...
    compile 'com.squareup.picasso:picasso:2.5.2'
}
```

```
Picasso.with(context)
     .load("http://servidor.com/foto.jpg")
     .resize(50, 50)
     .centerCrop()
     .placeholder(R.drawable.user_placeholder)
     .error(R.drawable.user_placeholder_error)
     .into(imageView);
```

Carregamento de imagens++



https://github.com/nostra13/Android-Universal-Image-Loader

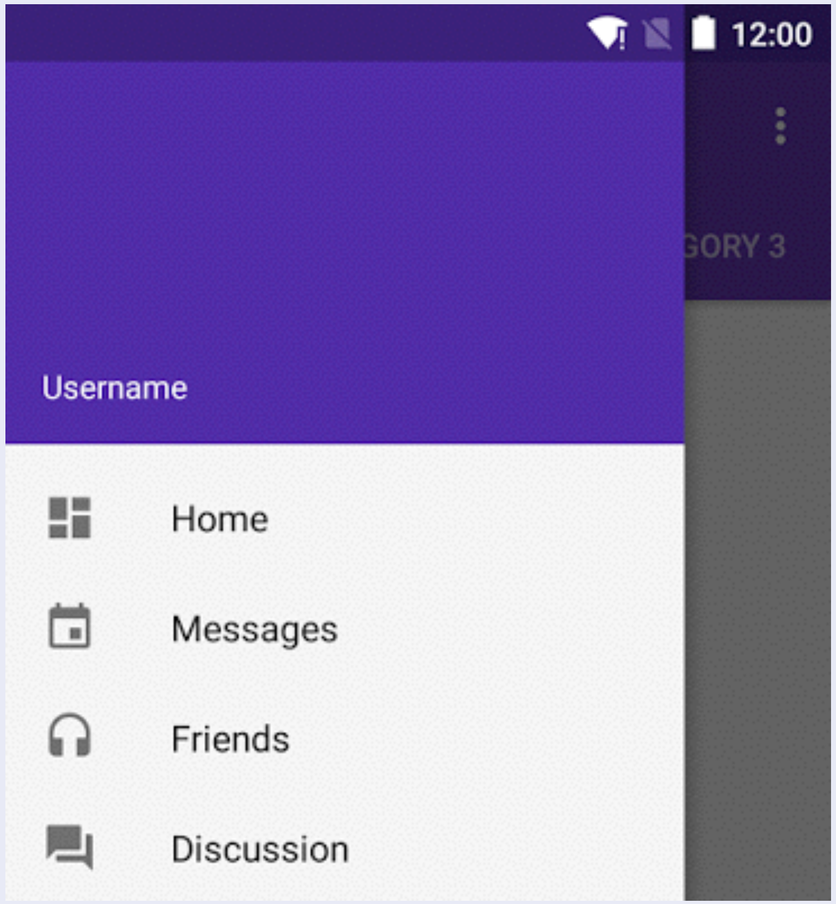
Colt McAnlis 8:21 AM *+1*

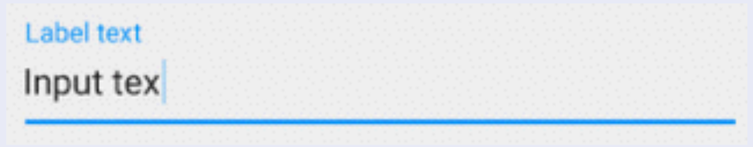
If you're doing *anything* with bitmaps, you should be using GLIDE. https://github.com/bumptech/glide It's going to provide the most memory efficient footprint that you can, alongside a ton of other bells and whistles.

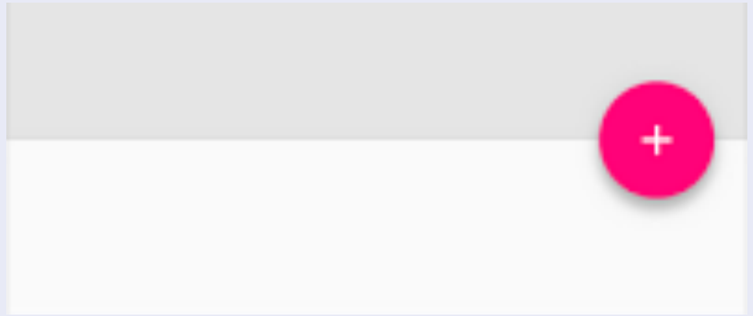
Design Support Library

- Lançada no Google I/O 2015.
- Traz diversas recomendações do Material Design prontas para usar.
- FAB, Tabs, Scroll, Navigation View, Snack Bar...

```
dependencies {
    ...
    compile 'com.android.support:design:22.2.0'
}
```

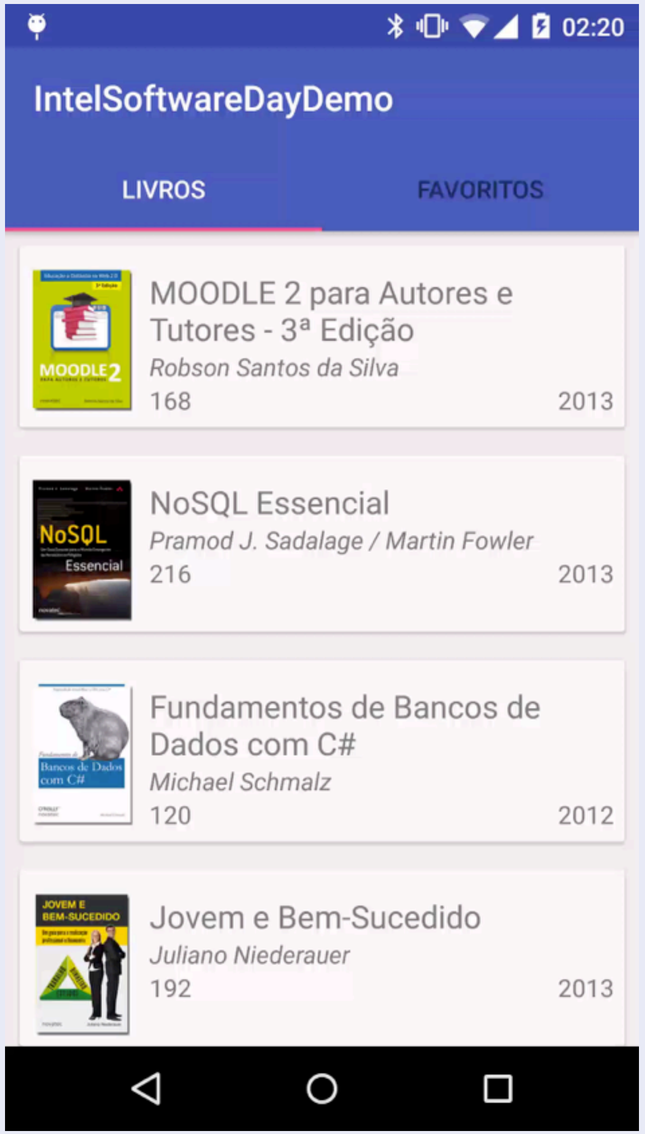

12:00
GORY 3
Username
Home
Messages
Friends
Discussion


Label text
Input tex

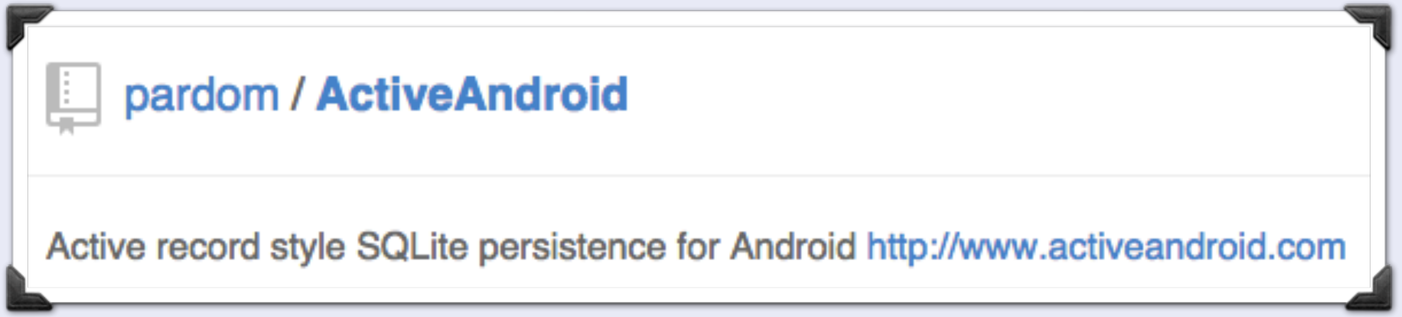



ITEM ONE
ITEM TWO
ITEM THREE

```
<android.support.design.widget.CoordinatorLayout ...>
    <android.support.v4.view.ViewPager ...
        app:layout_behavior="@string/appbar_scrolling_view_behavior" />
    <android.support.design.widget.AppBarLayout ...>
        <android.support.v7.widget.Toolbar ...
            app:layout_scrollFlags="scroll|enterAlways" />
        <android.support.design.widget.TabLayout .../>
    </android.support.design.widget.AppBarLayout>

</android.support.design.widget.CoordinatorLayout>
```

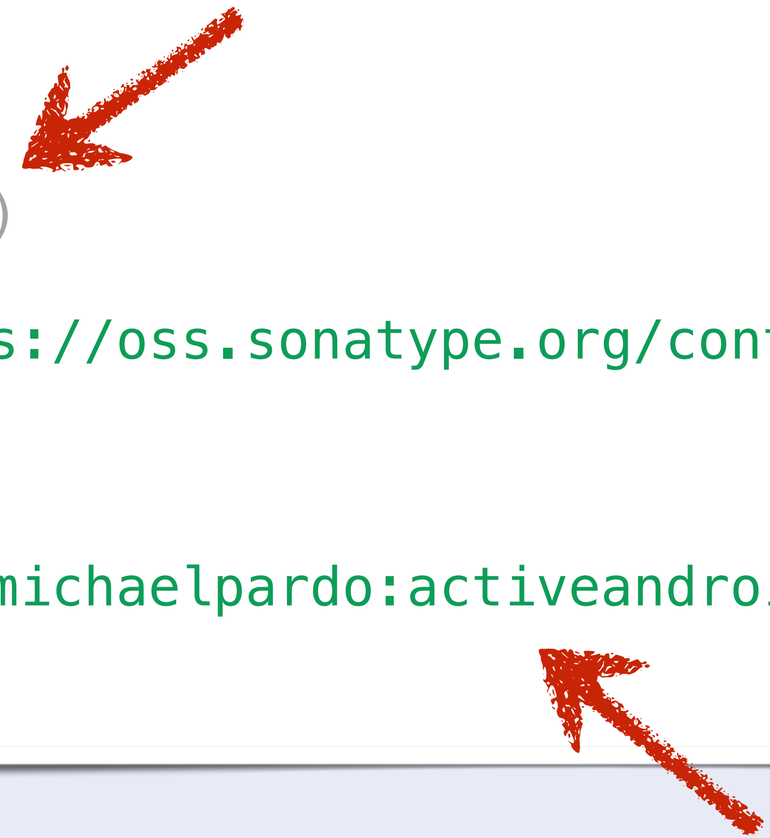

02:20
IntelSoftwareDayDemo
LIVROS
FAVORITOS
MOODLE 2 para Autores e Tutores - 3ª Edição
Robson Santos da Silva
168
2013
NoSQL Essencial
Pramod J. Sadalage / Martin Fowler
216
2013
Fundamentos de Bancos de Dados com C#
Michael Schmalz
120
2012
Jovem e Bem-Sucedido
Juliano Niederauer
192
2013

Persistência de dados



https://github.com/pardom/ActiveAndroid

```
apply plugin: 'com.android.application'

android {
  ...
  repositories {
    mavenCentral()
      maven {
        url "https://oss.sonatype.org/content/repositories/snapshots/" }
  }
  dependencies {
    ...
    compile 'com.michaelpardo:activeandroid:3.1.0-SNAPSHOT'
  }
}
```



```xml
<manifest ...>
    <application ...
        android:name=".App">
        ...
        <meta-data android:name="AA_DB_NAME"
                   android:value="livros.db"/>

        <meta-data android:name="AA_DB_VERSION"
                   android:value="1"/>
    </application>
</manifest>
```

```java
import android.app.Application;
import com.activeandroid.ActiveAndroid;

public class App extends Application {

    @Override
    public void onCreate() {
        super.onCreate();
        ActiveAndroid.initialize(this);
    }
}
```

```java
import com.activeandroid.Model;
import com.activeandroid.annotation.Column;
import com.activeandroid.annotation.Table;

@Table(name = "Livros")
public class Livro extends Model {
    @Column(name = "titulo", unique = true,
            onUniqueConflict = Column.ConflictAction.REPLACE)
    public String titulo;
    @Column(name = "autor")
    public String autor;
    @Column(name = "ano")
    public int ano;
    @Column(name = "paginas")
    public int paginas;
    @Column(name = "capa")
    public String capa;
}
```

```
Livro livroFavorito = new Livro(
        "Dominando o Android",
        "Nelson Glauber",
        2015,
        792,
        "http://goo.gl/capa.png"
);
livroFavorito.save();
```

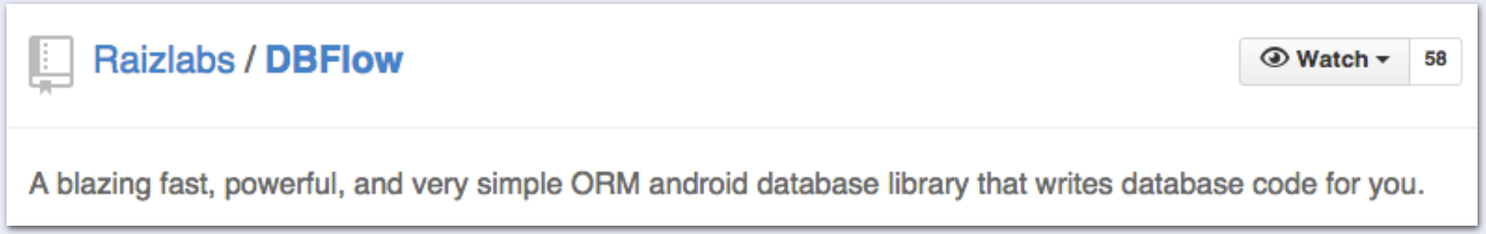
```
List<Livro> mLivros = new Select()
        .from(Livro.class)
        .where("titulo LIKE ?", "Dom%")
        .execute();
```

```
Livro livro = new Select()
                .from(Livro.class)
                .where("ID = ?", 12)
                .executeSingle();
```

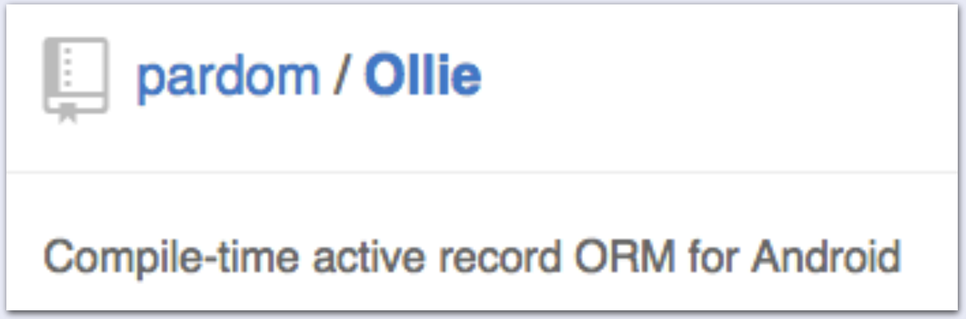
Mapeamento Objeto-Relacional (ORM)

- Compatível com **ContentProvider** e a classe **Cursor**.
- Suporte ao **onUpgrade()** através de arquivos SQL disponibilizados na pasta assets do projeto.
- Mais informações: https://guides.codepath.com/android/ActiveAndroid-Guide

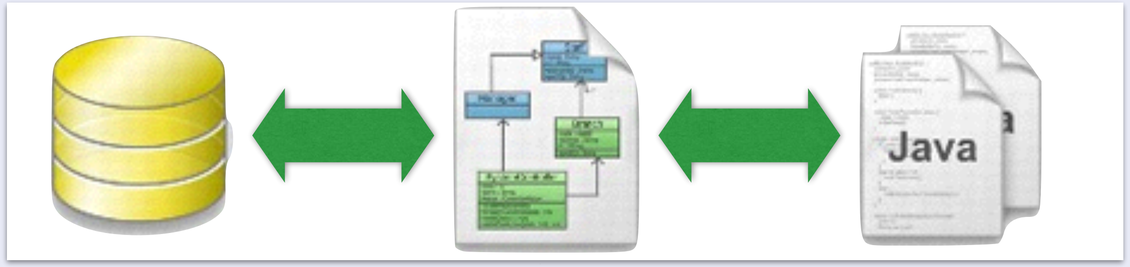
Persistência de dados++



https://github.com/Raizlabs/DBFlow



https://github.com/pardom/ollie



http://ormlite.com/sqlite_java_android_orm.shtml

Troca de Mensagens

Otto

An enhanced **event bus** with emphasis on Android support

http://square.github.io/otto/

```
dependencies {
    ...
    compile 'com.squareup:otto:1.3.5'
}
```

```
import android.app.Application;
import com.squareup.otto.Bus;

public class App extends Application {
    Bus mBus;

    @Override
    public void onCreate() {
        super.onCreate();
        ...
        mBus = new Bus();
    }
    public Bus getBus(){
        return mBus;
    }
}
```

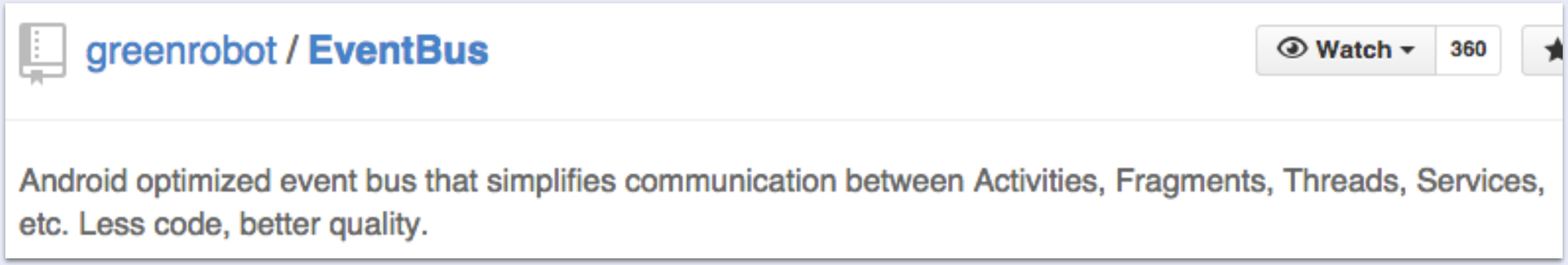
```
Bus mBus = ((App)getApplication()).getBus();

mBus.register(this);
```

```
mBus.post(new UmaClasseQualquer("Mensagem"));
```

```
@Subscribe
public void chegouEvento(UmaClasseQualquer event) {
    String texto = event.getMensagem();
    Toast.makeText(this, texto, Toast.LENGTH_LONG).show();
}
```

```
mBus.unregister(this);
```

Troca de Mensagens++



https://github.com/greenrobot/EventBus

Material icons

- Aprox. 800 ícones no padrão Material Design
- Gratuitos
- Para todas as densidades de tela
- **https://www.google.com/design/icons/**

Injeção de dependência

http://square.github.io/dagger/

```
dependencies {
    ...
    compile 'com.squareup.dagger:dagger:1.2.2'
    provided 'com.squareup.dagger:dagger-compiler:1.2.2'
}
```

Dagger

- O Dagger **constrói instâncias** das classes da aplicação e **satisfaz suas dependências**.
- Ele trabalha em **tempo de compilação** e não em tempo de execução. Tornando-o mais eficiente.
- Diminiu o acoplamento entre a UI e os "serviços".
- Deixa o código mais testável.

Declarando dependências

- Utilizamos **@Inject** para anotar um construtor ou atributo que o Dagger deve fornecer a dependência.
- Ou podemos fazer uma “lazy injection” usando: **@Inject Lazy<SeuServico>** servico;

```
public class ListaFavoritosFragment extends Fragment {

    @Inject
    Bus mBus;
    ...
```

```
public class ListaLivrosFragment extends Fragment {

    @Inject
    Lazy<LivroHttpService> livroService;
    ...
```

Satisfazendo dependências

- Para satisfazer as dependências devem ser criadas classes chamadas de módulos e anotadas com **@Module**.
- Métodos que criarão as instâncias devem estar anotados com **@Provides** e por convenção começam com “provides”.
- O **@Provides** pode ser usado em conjunto com **@Singleton**. É uma boa prática adicionar **@Singleton** na classe que será injetada para facilitar o entendimento.

```
import dagger.Module;
import dagger.Provides;

@Module(injects = {
        ListaFavoritosFragment.class,
        ListaLivrosFragment.class
})
public class AppModule {
    @Provides
    @Singleton
    public Bus provideBus() {
        return new Bus();
    }

    @Provides
    public LivroHttpService provideLivroHttpService() {
        return new LivroHttpService();
    }
}
```

ObjectGraph

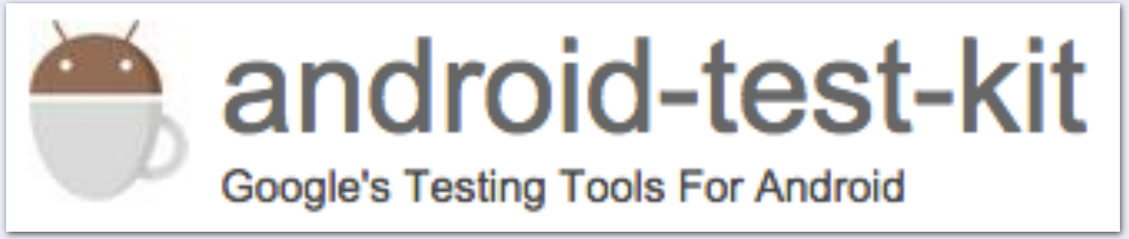
- **@Inject** + **@Provides** formam uma árvore de classes interligadas por suas dependências.

```
ObjectGraph objectGraph =
                    ObjectGraph.create(new AppModule());
objectGraph.inject(this);
```

Testes Unitários

http://mockito.org/

Testes de UI



https://code.google.com/p/android-test-kit/

Espresso

- Possui basicamente 3 componentes:
  - **ViewMatchers** para achar a view
  - **ViewActions** para realizar ações
  - **ViewAssertions** para checar o estado da view

```
onView(Matcher)
      .perform(ViewAction)
      .check(ViewAssertion)
```

CHEAT SHEET
espresso 2.0

Matchers

USER PROPERTIES
```
withId(…)
withText(…)
withTagKey(…)
withTagValue(…)
hasContentDescription(…)
withContentDescription(…)
withHint(…)
withSpinnerText(…)
hasLinks()
hasEllipsizedText()
hasMultilineText()
```

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

COMMON HAMCREST MATCHERS
```
allOf(Matchers)
anyOf(Matchers)
is(...)
not(...)
endsWith(String)
startsWith(String)
```

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

INPUT
```
supportsInputMethods(…)
hasImeAction(…)
```

CLASS
```
isAssignableFrom(…)
withClassName(…)
```

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

SEE ALSO
Preference matchers
Cursor matchers

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)
```

View Assertions

```
matches(Matcher)
doesNotExist()
selectedDescendantsMatch(…)
```

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

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

```
android {
  ...
  defaultConfig {
    ...
    testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
  }

  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’
    androidTestCompile ‘com.android.support.test.espresso:espresso-intents:2.2'
    androidTestCompile 'com.android.support.test.espresso:espresso-contrib:2.2'
  }
}
```

```java
import static android.support.test.espresso.Espresso.onView;
import static android.support.test.espresso.Espresso.pressBack;
import static android.support.test.espresso.action.ViewActions.click;
import static android.support.test.espresso.action.ViewActions.swipeLeft;
import static android.support.test.espresso.assertion.ViewAssertions.matches;
import static android.support.test.espresso.matcher.ViewMatchers.hasDescendant;
import static android.support.test.espresso.matcher.ViewMatchers.withId;
import static android.support.test.espresso.matcher.ViewMatchers.withText;

@RunWith(AndroidJUnit4.class)
@LargeTest
public class EspressoTest extends ActivityInstrumentationTestCase2<MainActivity> {

    public EspressoTest() {
        super(MainActivity.class);
    }

    @Before
    public void setUp() throws Exception {
        super.setUp();
        injectInstrumentation(InstrumentationRegistry.getInstrumentation());
    }
    // seus testes aqui...
}
```

```
@Test
public void testAdicionarFavorito() {
    String bookTitle = "NoSQL Essencial";
    onView(withId(R.id.recyclerViewWeb))
            .perform(RecyclerViewActions.actionOnItem(
                    hasDescendant(withText(bookTitle)), click()));
    onView(withId(R.id.txtTitulo)).check(matches(withText(bookTitle)));
    onView(withId(R.id.action_favorito)).perform(click());
    pressBack();
    onView(withId(R.id.pager)).perform(swipeLeft());
    onView(withId(R.id.recyclerViewFavoritos))
            .check(matches(hasDescendant(withText(bookTitle))));
}
```

Testes de UI++

https://code.google.com/p/robotium/

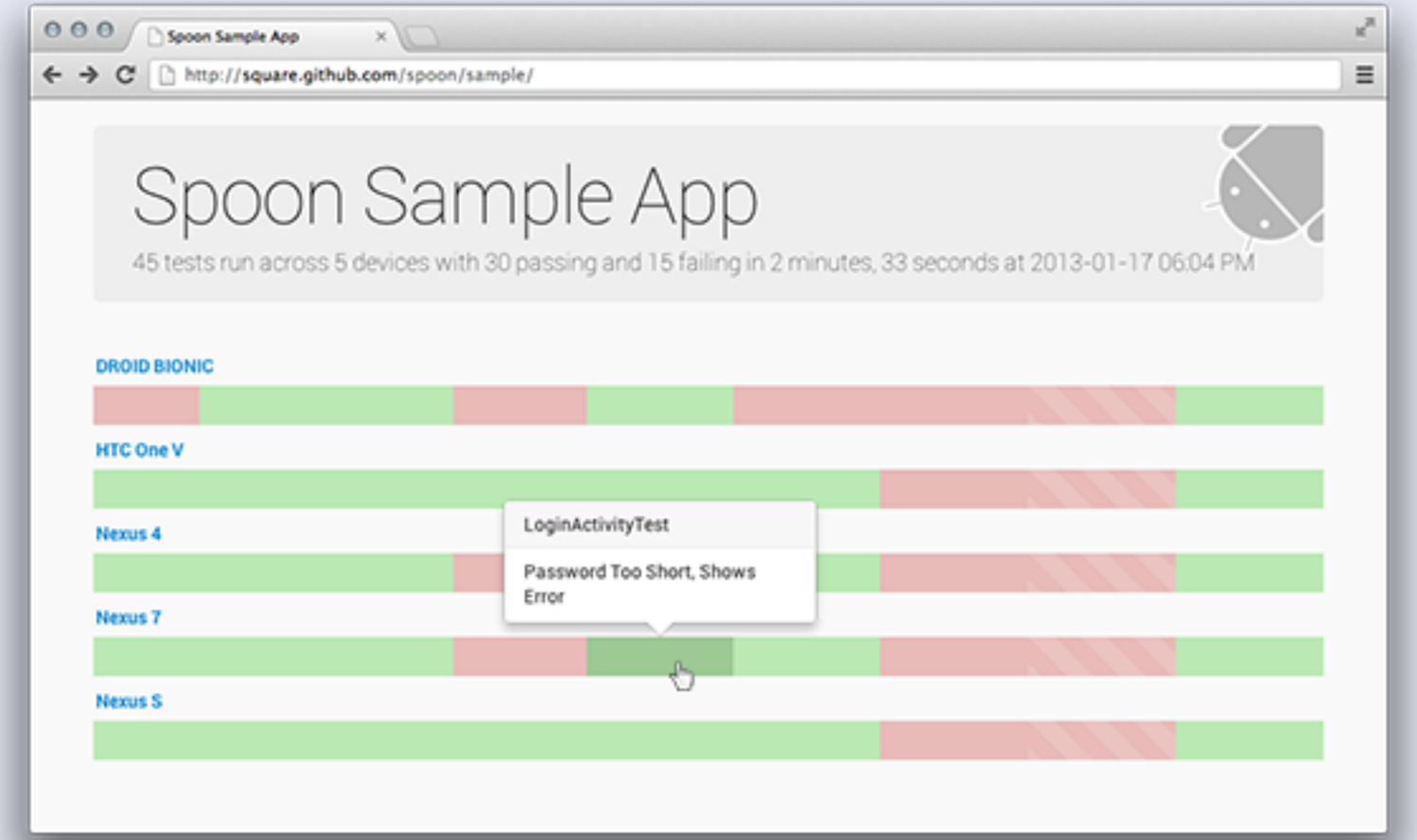
Execução de Testes

Spoon

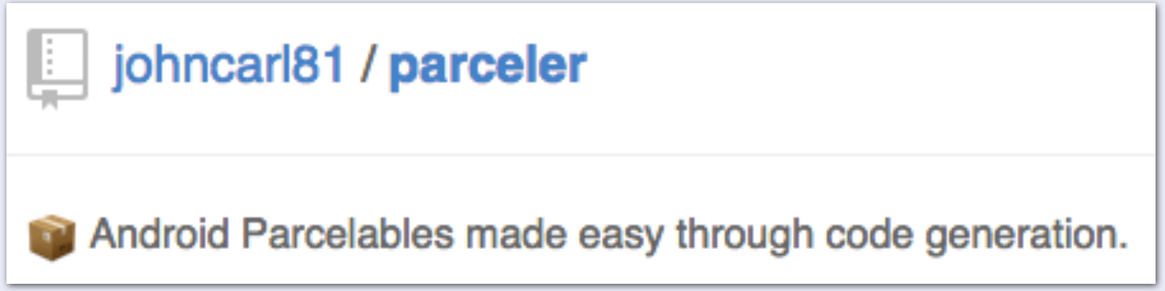
Distributing **instrumentation tests** to all your Androids

http://square.github.io/spoon/

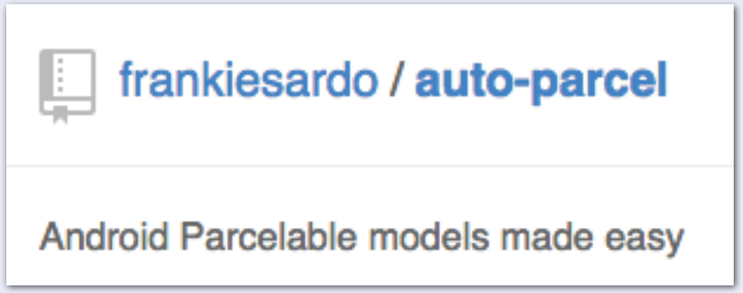
```
buildscript {
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath 'com.stanfy.spoon:spoon-gradle-plugin:0.14.1'
    }
}

apply plugin: 'spoon'

// This section is optional
spoon {
    // for debug output
    debug = true

    // To run a single test class
    className = 'br.com.nglauber.explorandolibs.TesteMainActivity'

    // To run a single method in TestCase
    methodName = 'testSimplesClick'
}
```


Spoon Sample App
http://square.github.com/spoon/sample/
Spoon Sample App
45 tests run across 5 devices with 30 passing and 15 failing in 2 minutes, 33 seconds at 2013-01-17 06:04 PM
DROID BIONIC
HTC One V
Nexus 4
Nexus 7
Nexus S
LoginActivityTest
Password Too Short, Shows Error

Diversos (Parcelable)



https://github.com/johncarl81/parceler



https://github.com/frankiesardo/auto-parcel

```
dependencies {
    ...
    compile 'org.parceler:parceler-api:0.2.15'
    provided 'org.parceler:parceler:0.2.15'
}
```

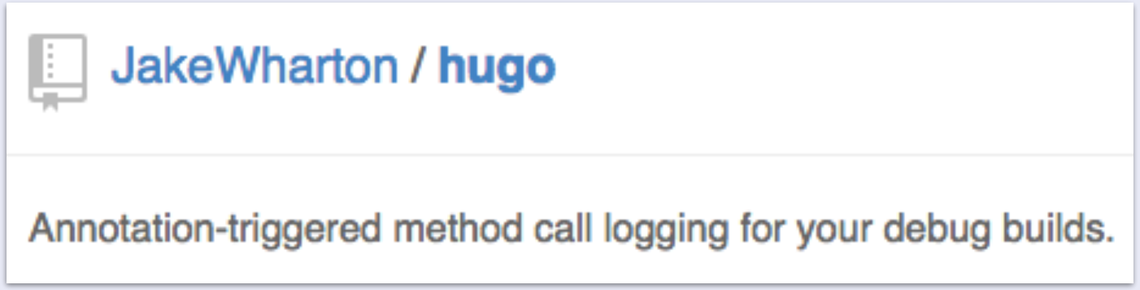
```
import org.parceler.Parcel;

@Parcel
public class Pessoa {
    String nome;
    int idade;

    public Pessoa(){ // obrigatório }

    public Pessoa(int i, String n) {
        this.idade = i;
        this.nome = n;
    }

    public String getNome() { return nome; }

    public int getIdade() { return idade; }
}
```

```
Parcelable wrapped = Parcels.wrap(new Pessoa(31, "Glauber"));
```

```
Pessoa pessoa = Parcels.unwrap(wrapped);
```

Log

https://github.com/JakeWharton/timber



https://github.com/JakeWharton/hugo

```
dependencies {
    ...
    compile 'com.jakewharton.timber:timber:2.5.1'
}
```

```
// No onCreate da App...
if (BuildConfig.DEBUG) {
    Timber.plant(new Timber.DebugTree());
} else {
    Timber.plant(new CrashReportingTree());
}
```

```
Timber.d("Usando o Timber!");
```

```
class CrashReportingTree extends Timber.HollowTree {
    @Override
    public void i(String message, Object... args) {
    }

    @Override
    public void i(Throwable t, String message, Object... args) {
      i(message, args);
    }

    @Override
    public void e(String message, Object... args) {
      i("ERROR: " + message, args);
      // Enviar log para o servidor
    }

    @Override
    public void e(Throwable t, String message, Object... args) {
      e(message, args);
      // Enviar log para o servidor
    }
}
```

Analytics

http://try.crashlytics.com/sdk-android/

https://developers.google.com/analytics/devguides/collection/android/v4/

Diversos

- **Calligraphy** - Para fontes customizadas
  https://github.com/chrisjenx/Calligraphy
- **Joda-Time** - Para trabalhar com data e hora
  https://github.com/dlew/joda-time-android
- **AndroidViewAnimations** - Animações pré-definidas
  https://github.com/daimajia/AndroidViewAnimations
- **seismic** - Para shake detection
  https://github.com/square/seismic

while(true) diversos++;

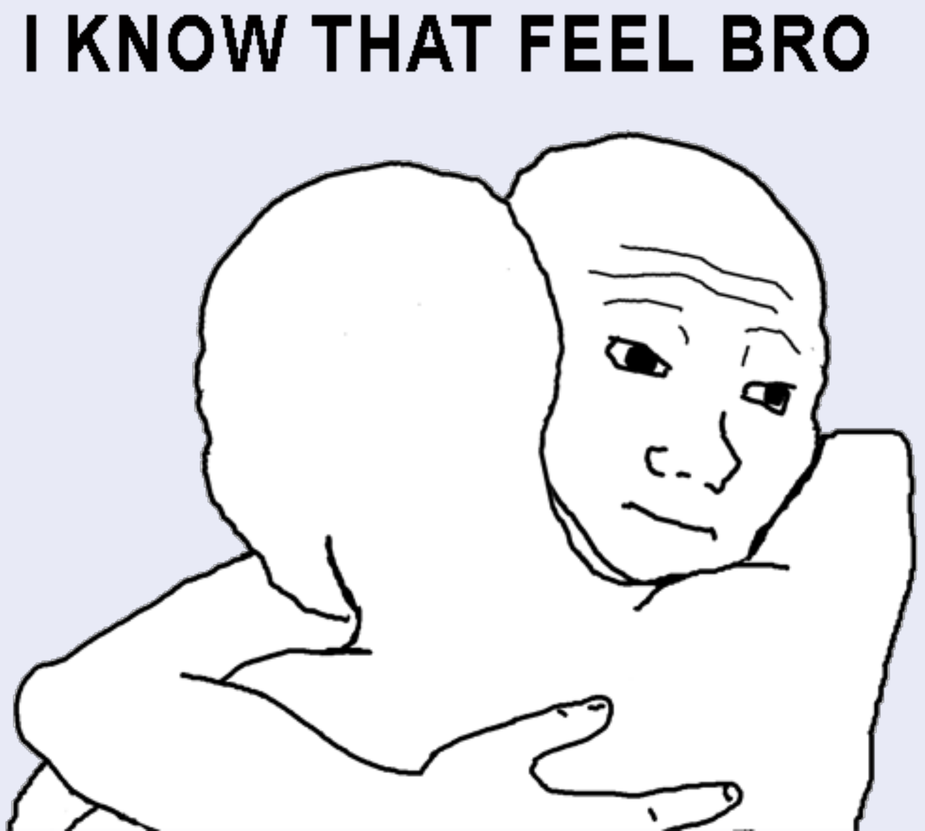
https://android-arsenal.com/

Vou utilizar todas no meu projeto!!!

É importante conhecer (e bem) a API padrão para saber onde você ganhará produtividade usando uma lib

Não esqueça da licença!!!

E lembre-se! A API padrão nunca vai te abandonar!



Avalie essa palestra!
http://www.bit.ly/ISD_Pesquisa

Demo
https://github.com/nglauber/intelsoftwareday2015

@nglauber

+NelsonGlauber

www.nglauber.com.br


Dúvidas?
Obrigado!!