
DEVELOPING IN ANDROID
(LA PELÍCULA)
UNA PELÍCULA DE
JOSÉ MANUEL PEREIRA

¿QUIEN? JOSÉ MANUEL PEREIRA

jm.pereira.g@gmail.com

https://github.com/JMPergar

https://plus.google.com/+JoséMPereira

www.androcode.es

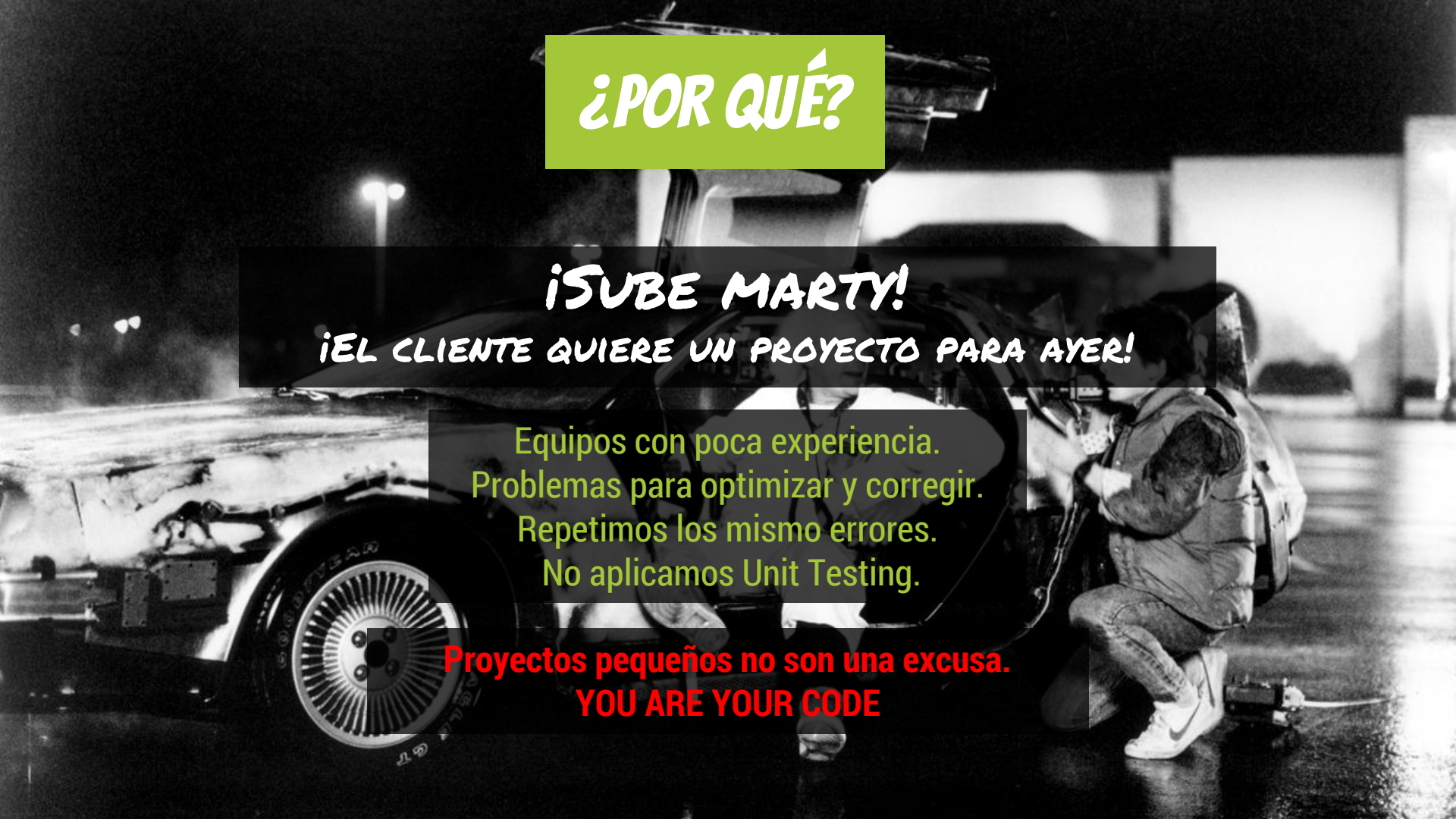
@JMPergar

http://www.slideshare.net/jmpereirag

https://www.linkedin.com/in/jmpergar

¿QUE HAGO?

- Android Developer
- Mobile Team Lead at BeRepublic
- Member of Core Team at GDG Barcelona



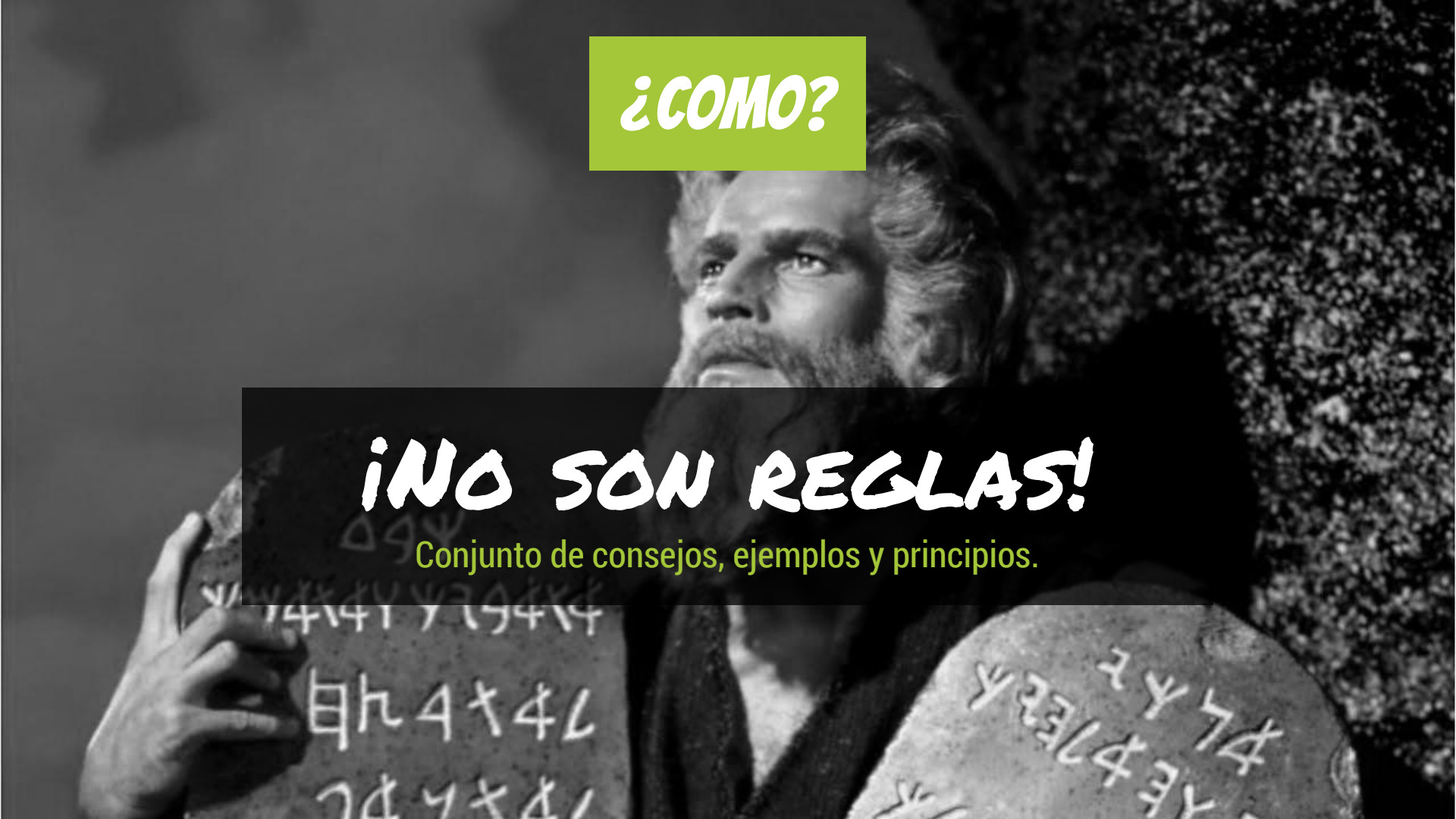
¿POR QUÉ?

¡SUBE MARTY!

¡EL CLIENTE QUIERE UN PROYECTO PARA AYER!

Equipos con poca experiencia.
Problemas para optimizar y corregir.
Repetimos los mismo errores.
No aplicamos Unit Testing.

Proyectos pequeños no son una excusa.
YOU ARE YOUR CODE



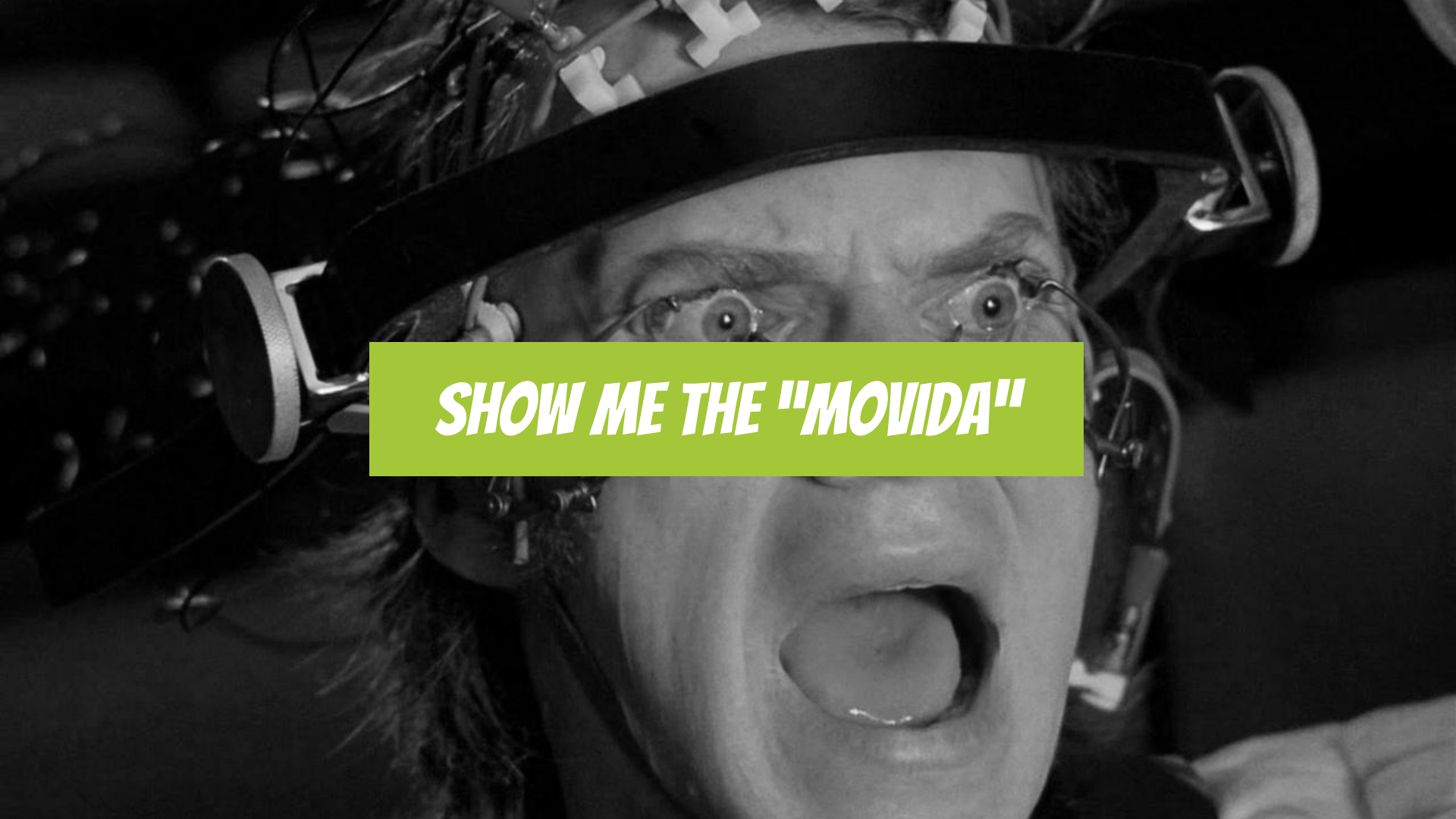
¿COMO?

¡NO SON REGLAS!

Conjunto de consejos, ejemplos y principios.


¿COMO?
KAIZEN 改善
Mejora continua.
Done is better than perfect.
Si tras 6 meses tu código no te da vergüenza ajena no lo estas haciendo bien. (visto en twitter)


¿COMO?
YO HE VISTO COSAS QUE VOSOTROS NO CREERÍAIS
Básico pero necesario.


SHOW ME THE "MOVIDA"

NAMMING

Mi nombre es Iñigo Montoya

¡El que sea, pero aplica uno!
Identifica actores del framework.
Identifica patrones.
Respeta los nombres comunes y crea los tuyos.
Aplicalo en todos los niveles.
Muy importante en los recursos.

NAMMING

Antes

Después

NAMMING

Drawables

group_type_name_state_suffix

→ actionbar_icon_create_disabled, common_background_app

Layouts

type_name_suffix

→ activity_login, fragment_profile, adapter_user, include_header_premium

Dimens

property_default_group_type_name

→ fontsize_default, height_common_button

Id's

type_name

→ cv_footer, tv_name, iv_avatar

Classes

NameBaseType

→ BaseActivity, ProfileFragment, ScreenUtils, RenderFactory, UserMVO, PostDAO

Commons names

colors.xml, config.xml, dimens.xml, strings.xml, plurals.xml, arrays.xml, styles.xml, themes.xml...

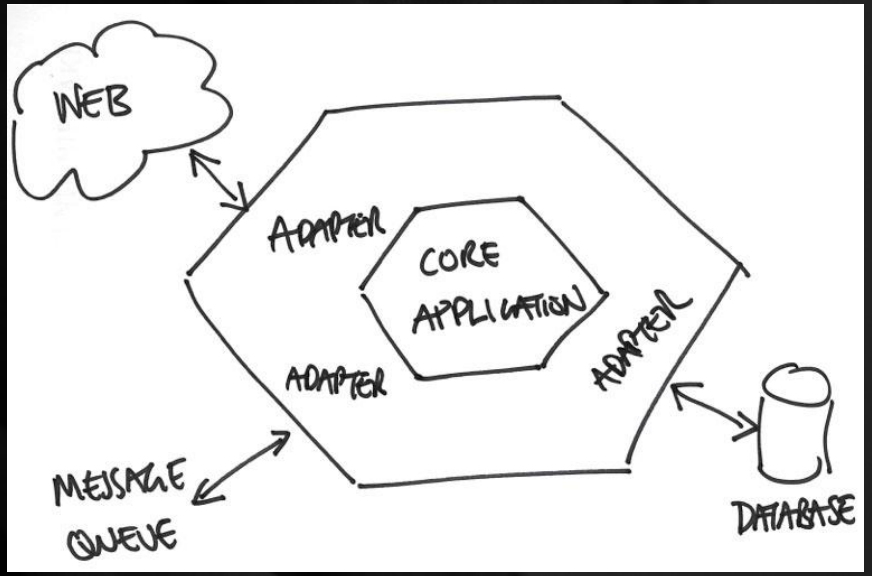
PACKAGING

What's in the box??

~~¡El que sea, pero aplica uno!~~
Básico para ser organizado.
Es la base de nuestras arquitecturas.

PACKAGING

- src
  - main
    - java
      - com
        - berepublic
          - structure
            - util
          - desigual
            - desigual360
              - model
                - api
                  - response
                  - Desigual360API
                  - Desigual360APIManager
                - boundary
                - presenter
                  - dto
                  - CatwalkModel
                  - CatwalkPresenter
                  - CatwalkView
                  - HomeModel
                  - HomePresenter
                  - HomeView
                - view
                  - activity
                  - adapter
                  - boundary
                  - fragment
                  - mvo
                  - view
                - BaseApplication
    - res



ARCHITECTURE

MVC, MVP, Clean Architecture,
Ports and Adapters...

Usa la arquitectura que quieras
pero aplica S.O.L.I.D.
Barroso dixit

Te permitira aplicar testing unitario
https://www.youtube.com/watch?v=I0qDmbwGz3o [Fernando Cejas]
Te conducirá a aplicar patrones
https://www.youtube.com/watch?v=tt3zI9cKiWU [Pedro Vicente]
Hara tu app mas solida y escalable.
https://www.youtube.com/watch?v=ROdIvrLL1ao [Jorge Barroso]


S.O.L.I.D.
The Single responsibility principle
The Open closed principle
The Liskov substitution principle
The Interface segregation principle
The Dependency inversion principle

S.O.L.I.D.

Principio de Responsabilidad Única

"Una clase debería tener una y solo una razón para cambiar"
Robert C. Martin

Un objeto debe tener una única responsabilidad.

~~The God Activity~~

S.O.L.I.D.

Principio Abierto / Cerrado

Todo módulo debe estar abierto
para la extensión pero cerrado
para la modificación.

~~El Adapter pintalotodo~~

S.O.L.I.D.

Principio de Sustitución de Liskov

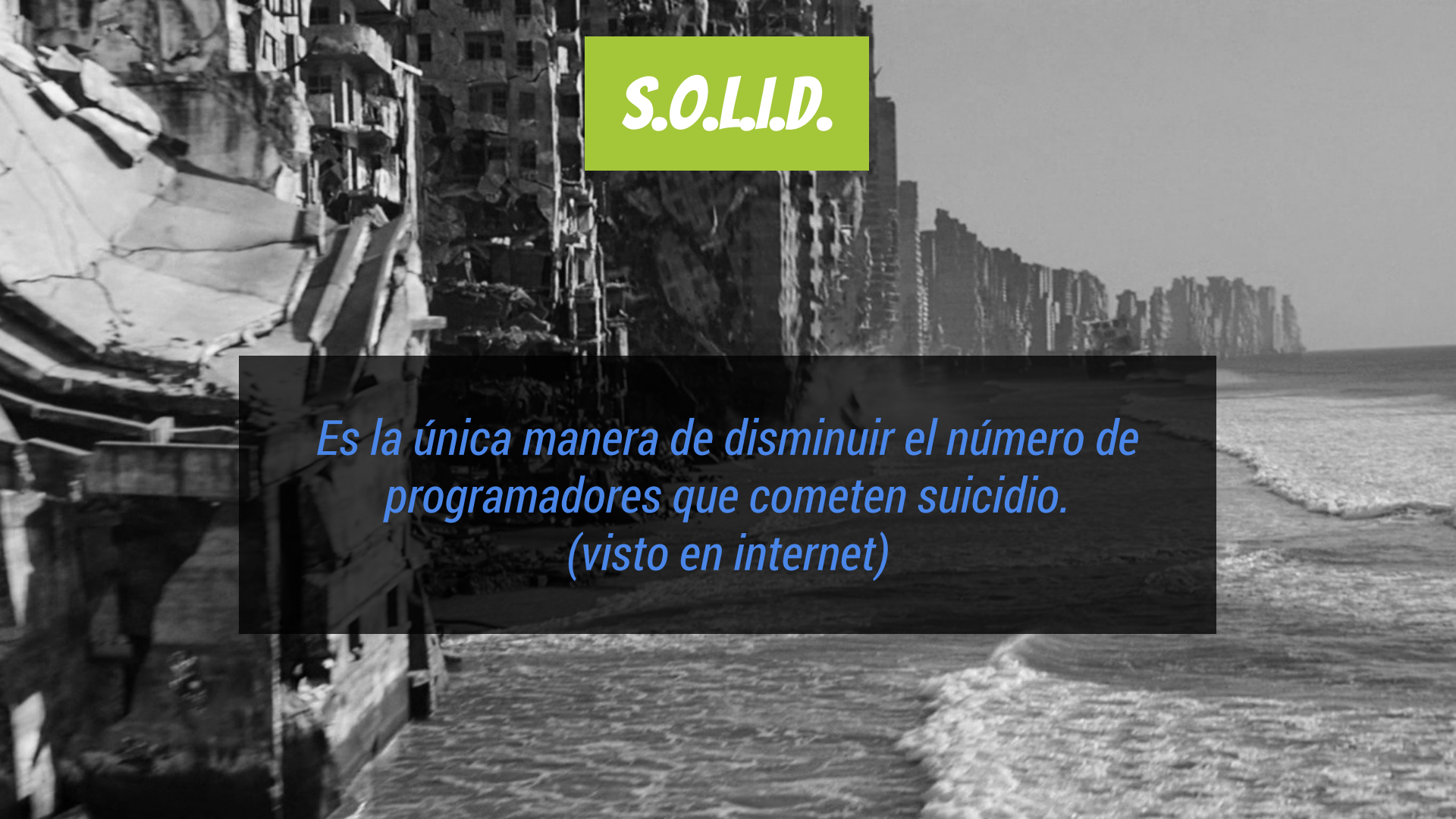
"Si parece un pato y grazna como un pato
pero necesita pilas,
probablemente no sea un pato."

"Los objetos de un programa deben poder reemplazarse por instancias de sus subtipos sin alterar la correctitud del programa"

Context


S.O.L.I.D.
Principio de Segregación de Interfaces
"Los clientes no deben ser forzados a depender de interfaces que no necesitan"
Robert C. Martin
Es preferible muchas interfaces específicas de cliente que una interfaz de uso general.
ViewPager.OnPageChangeListener


S.O.L.I.D.
Principio de Inversión de Dependencias
Debemos depender de las abstracciones
y no de las concreciones.
Api de datos

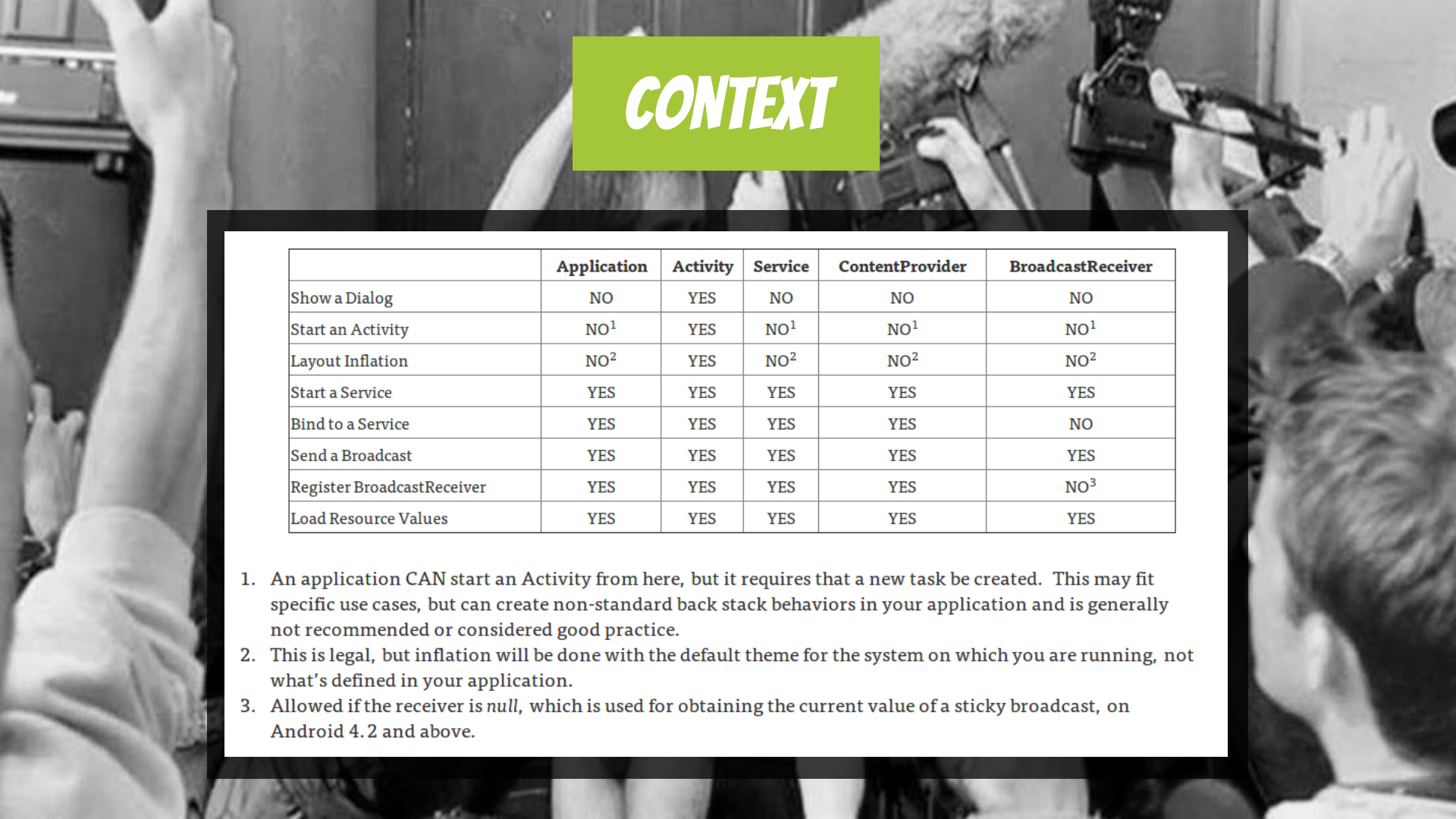

S.O.L.I.D.
Es la única manera de disminuir el número de programadores que cometen suicidio.
(visto en internet)

FRAGMENTATION AND THE FRAMEWORK

Resolución y densidades
Hardware
Versión de Sistema Operativo
Modificaciones de fabricantes
Forks de android

Desacoplar del framework es parte de la solución


CONTEXT
Context es probablemente el elemento más usado en el desarrollo de aplicaciones Android...
y quizás también el peor usado.
Application
Activity
Service
BroadcastReceiver
ContentProvider



CONTEXT

	Application	Activity	Service	ContentProvider	BroadcastReceiver
Show a Dialog	NO	YES	NO	NO	NO
Start an Activity	NO[1]	YES	NO[1]	NO[1]	NO[1]
Layout Inflation	NO[2]	YES	NO[2]	NO[2]	NO[2]
Start a Service	YES	YES	YES	YES	YES
Bind to a Service	YES	YES	YES	YES	NO
Send a Broadcast	YES	YES	YES	YES	YES
Register BroadcastReceiver	YES	YES	YES	YES	NO[3]
Load Resource Values	YES	YES	YES	YES	YES

1. An application CAN start an Activity from here, but it requires that a new task be created. This may fit specific use cases, but can create non-standard back stack behaviors in your application and is generally not recommended or considered good practice.
2. This is legal, but inflation will be done with the default theme for the system on which you are running, not what's defined in your application.
3. Allowed if the receiver is *null*, which is used for obtaining the current value of a sticky broadcast, on Android 4.2 and above.

CONTEXT

Bad Singleton

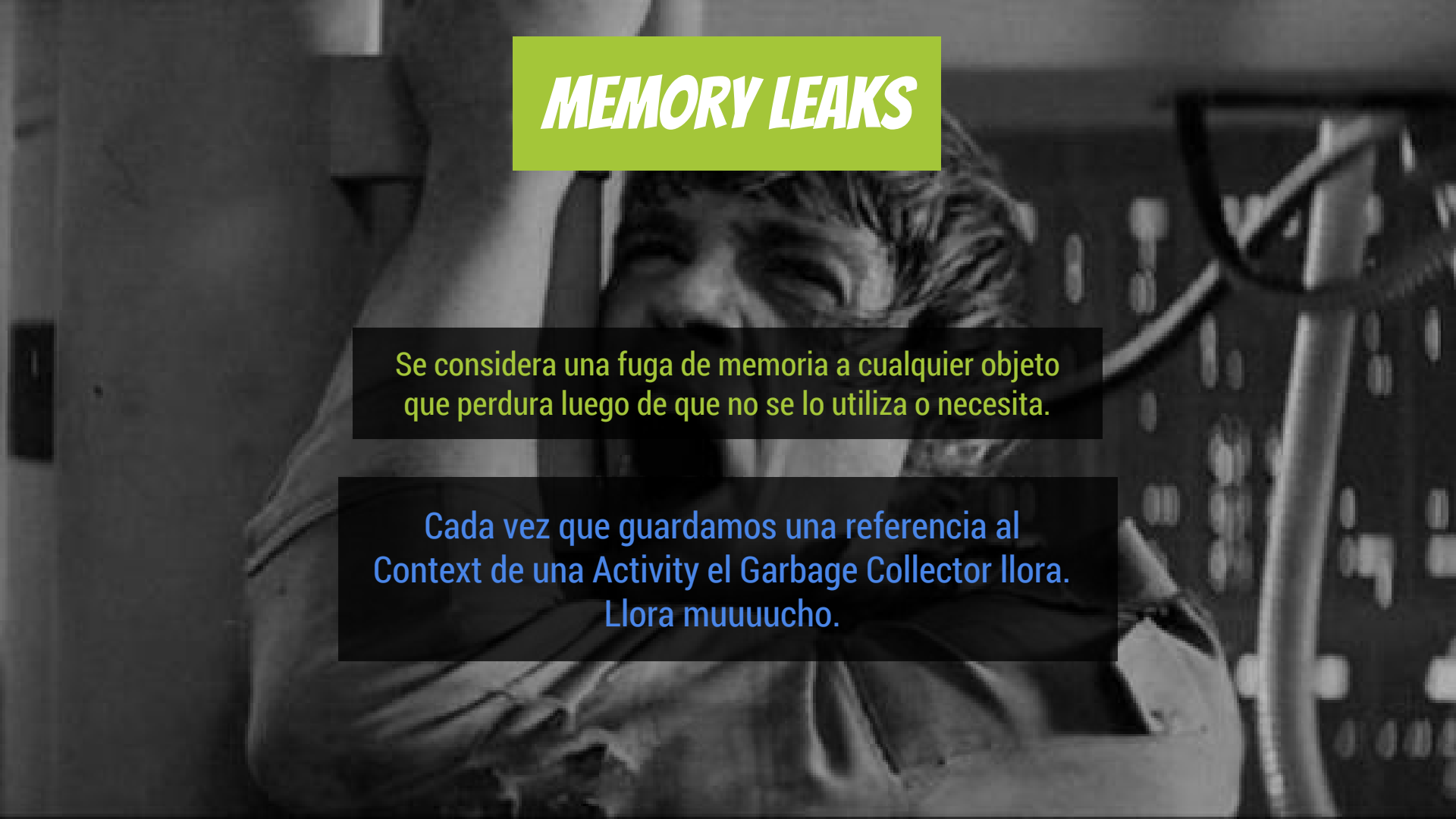
```
1  public class CustomManager {
2      private static CustomManager sInstance;
3
4      public static CustomManager getInstance(Context context) {
5          if (sInstance == null) {
6              sInstance = new CustomManager(context);
7          }
8
9          return sInstance;
10     }
11
12     private Context mContext;
13
14     private CustomManager(Context context) {
15         mContext = context;
16     }
17 }
```

CONTEXT

Better Singleton

```
1  public class CustomManager {
2      private static CustomManager sInstance;
3
4      public static CustomManager getInstance(Context context) {
5          if (sInstance == null) {
6              //Always pass in the Application Context
7              sInstance = new CustomManager(context.getApplicationContext());
8          }
9
10         return sInstance;
11     }
12
13     private Context mContext;
14
15     private CustomManager(Context context) {
16         mContext = context;
17     }
18 }
```

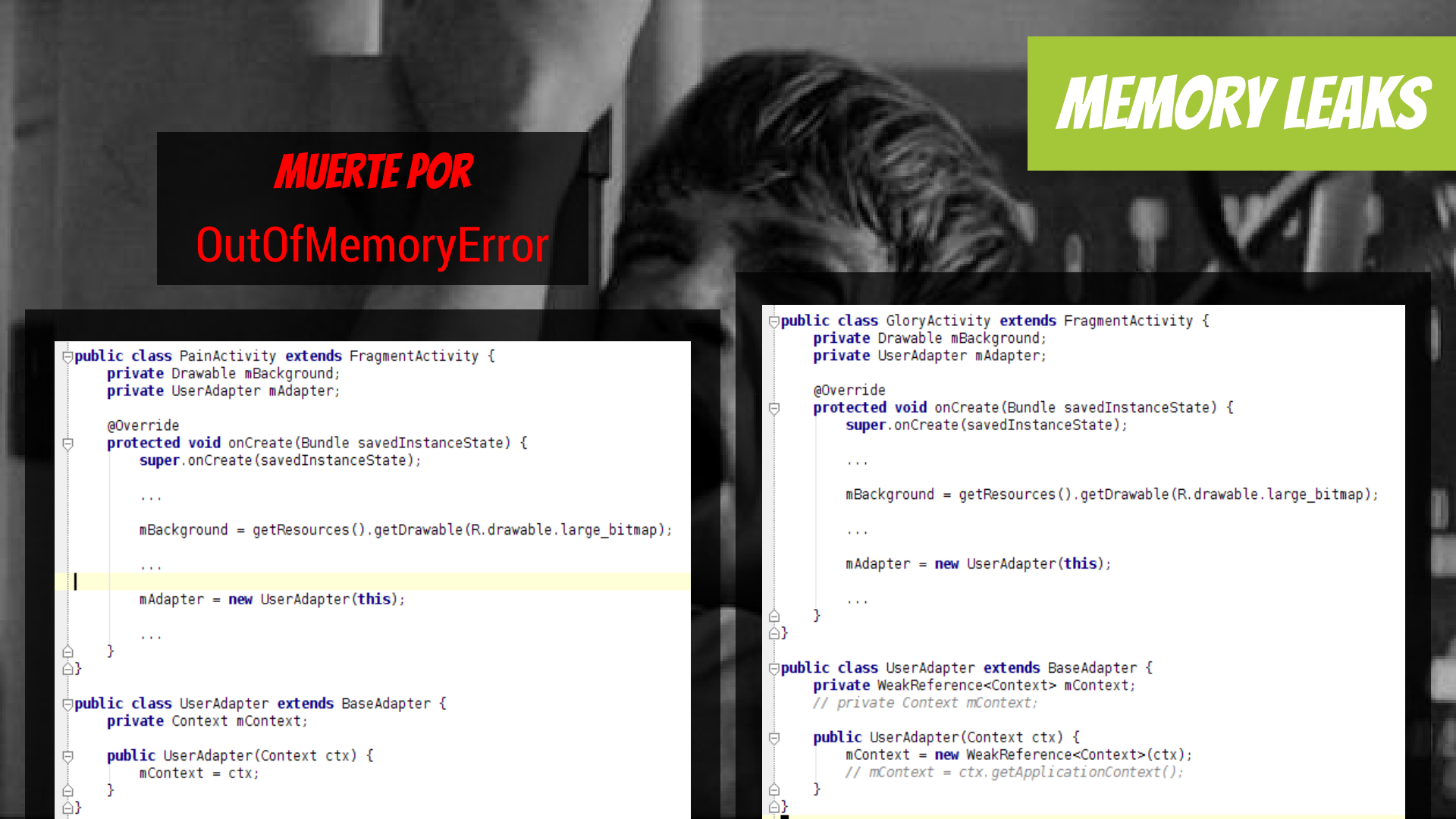

CONTEXT
más información en
Context, What Context?
http://www.doubleencore.com/2013/06/context/



MEMORY LEAKS

Se considera una fuga de memoria a cualquier objeto que perdura luego de que no se lo utiliza o necesita.

Cada vez que guardamos una referencia al Context de una Activity el Garbage Collector llora. Llora muuuucho.




```java
public class PainActivity extends FragmentActivity {
    private Drawable mBackground;
    private UserAdapter mAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        ...

        mBackground = getResources().getDrawable(R.drawable.large_bitmap);

        ...

        mAdapter = new UserAdapter(this);

        ...
    }
}

public class UserAdapter extends BaseAdapter {
    private Context mContext;

    public UserAdapter(Context ctx) {
        mContext = ctx;
    }
}
```

```java
public class GloryActivity extends FragmentActivity {
    private Drawable mBackground;
    private UserAdapter mAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        ...

        mBackground = getResources().getDrawable(R.drawable.large_bitmap);

        ...

        mAdapter = new UserAdapter(this);

        ...
    }
}

public class UserAdapter extends BaseAdapter {
    private WeakReference<Context> mContext;
    // private Context mContext;

    public UserAdapter(Context ctx) {
        mContext = new WeakReference<Context>(ctx);
        // mContext = ctx.getApplicationContext();
    }
}
```


MEMORY LEAKS
No guardar referencias al context-activity
Trata de usar context-application en lugar de context-activity
Usa WeakReference cuando no tengas más remedio que guardar las referencias.


FRONT-END
Layouts, Dimens,
style, themes,
colors, animations...
¡USALOS!

FRONT-END

INCLUDE

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/app_bg"
    android:gravity="center_horizontal">

    <include layout="@layout/titlebar"/>

    <TextView android:layout_width="match_parent"
              android:layout_height="wrap_content"
              android:text="@string/hello"
              android:padding="10dp" />

    ...

</LinearLayout>
```

FRONT-END

MERGE

```
<merge xmlns:android="http://schemas.android.com/apk/res/android">

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/add"/>

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/delete"/>

</merge>
```

FRONT-END

ViewStub

```
<ViewStub
    android:id="@+id/stub_import"
    android:inflatedId="@+id/panel_import"
    android:layout="@layout/progress_overlay"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom" />
```

```
((ViewStub) findViewById(R.id.stub_import)).setVisibility(View.VISIBLE);
// or
View importPanel = ((ViewStub) findViewById(R.id.stub_import)).inflate();
```

GRADDLE IS COMMING

Build Types

Flavors

Gestión de dependencias

Flavors Groups

GOOGLE+

http://goo.gl/2zgvlp

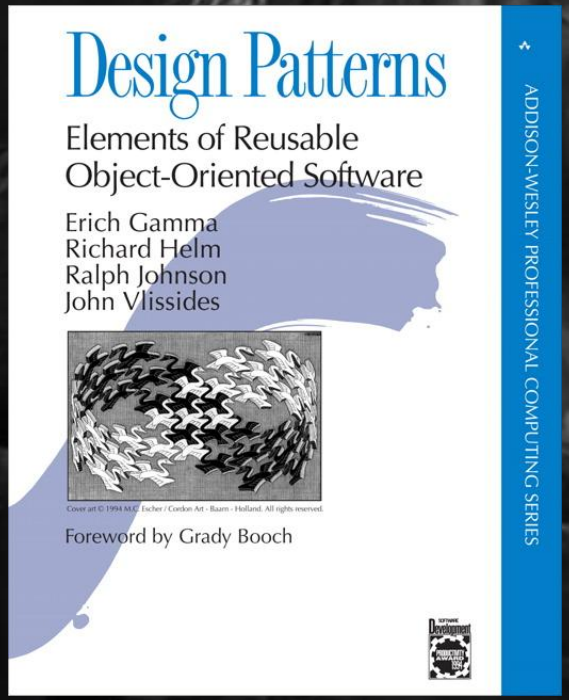
REFERENCIAS

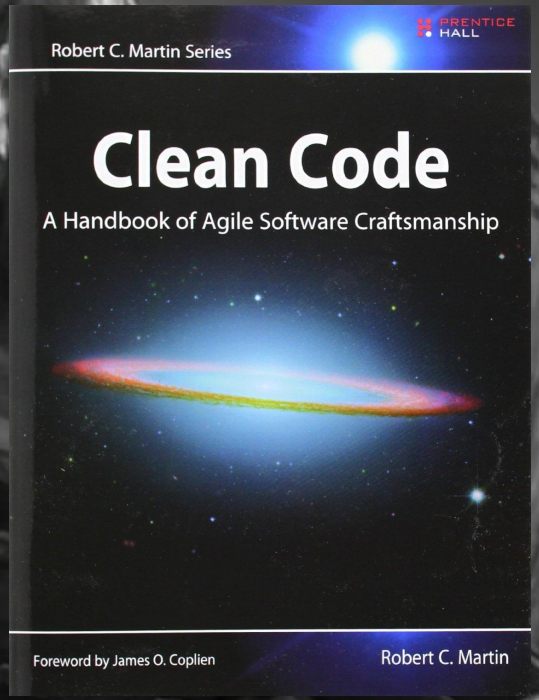
The CommonsBlog http://commonsware.com/blog/
sgoliver.net blog http://www.sgoliver.net/blog/?page_id=3011
Cyril Mottier http://cyrilmottier.com/
Dan Lew Codes http://blog.danlew.net/
Antonio Leiva http://antonioleiva.com/
ANDROID TALES http://android.amberfog.com/
Android Coding http://android-coding.blogspot.com.es/
Styling Android http://blog.stylingandroid.com/
Android Weekly http://androidweekly.net/
vogella.com http://www.vogella.com/tutorials/android.html
double encore http://www.doubleencore.com/tag/android/
Android-er http://android-er.blogspot.com.es/
Youtube: Android Developers https://www.youtube.com/user/androiddevelopers
AndroCode http://androcode.es/
Android Developers Blog http://android-developers.blogspot.com.es/
Grokking Android http://www.grokkingandroid.com/
ANDROID DESIGN PATTERNS http://www.androiddesignpatterns.com/

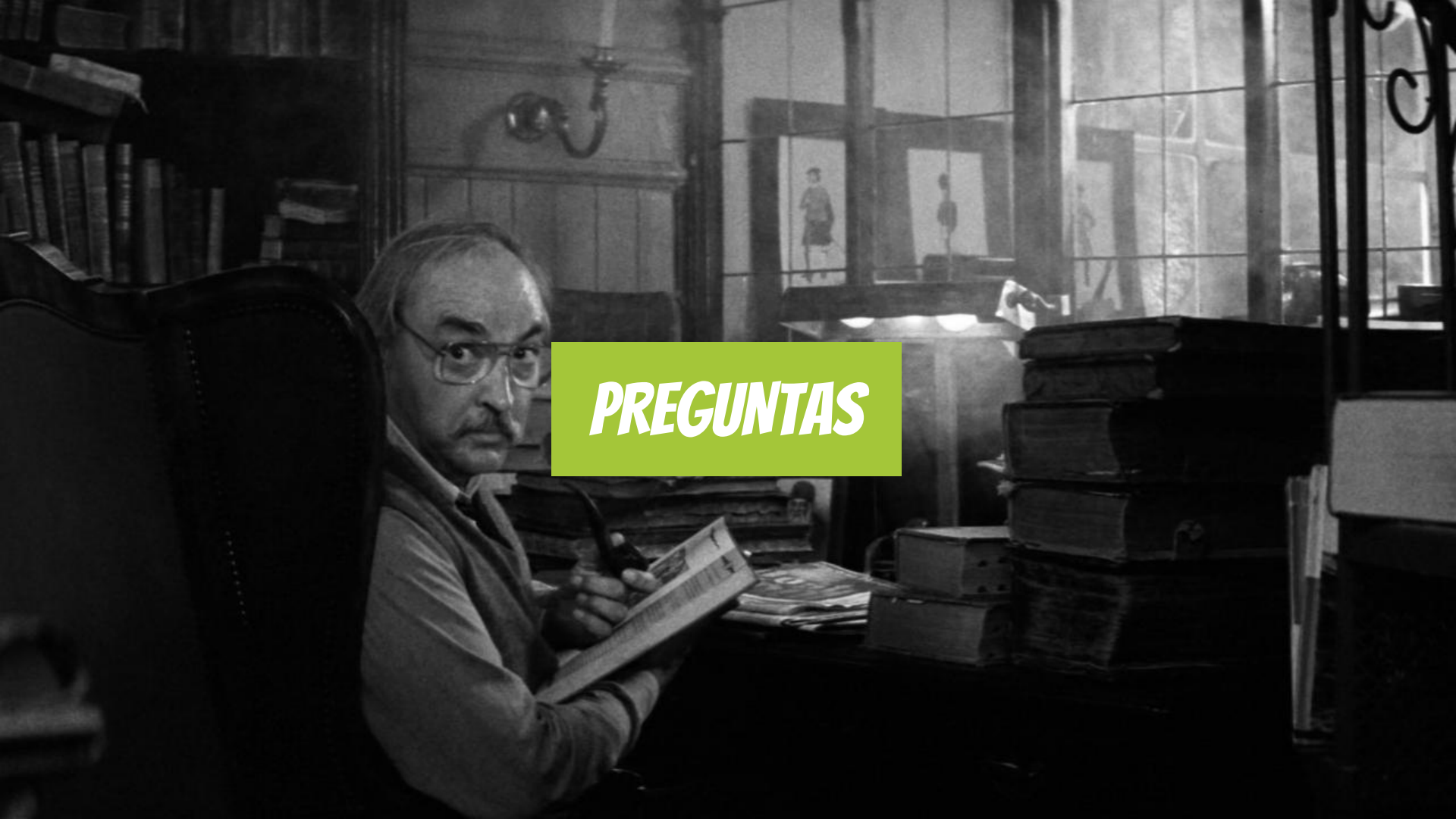
Twitter List https://twitter.com/JMPergar/android-dev-must/members

Android Arsenal http://android-arsenal.com/
AndroidViews http://www.androidviews.net/
Square Code Styles http://goo.gl/yZqppi

REFERENCIAS






PREGUNTAS

¡GRACIAS!