

# Poniendonos serios con Kivy

Sofía Martin y Lucas Chiacchio  
Facultad de Informática - UNLP

Noviembre 2016



# Contenido

## 1 Introducción

- ¿Qué es?
- Antes que nada

## 2 Entendiendo cómo se usa

- Estructura
- Comunicación

## 3 Diseño visual

- Agregar color a label y botones

# Características

- Librería UI Python orientada app multitouch.
- Motor gráfico basado en OpenGL ES 2.
- 100 % libre.
- Widgets: bloques GUI pre armados.
- Versión actual:

# Plataformas



# Instalación

- Instalar kivy:

```
git clone http://github.com/kivy/kivy
```

- Instalar y configurar algunas de todas las formas para compilar.

# Opciones compilación

- Usando el proyecto python-for-android.
- Imagen virtual ya armada con Kivy Android.
- Utilizando la herramienta **Buildozer**.
- Kivy Launcher.

# Kivy Launcher

- No se necesita compilar.
- Bajar al app Kivy Launcher.
- Copiar a /sdcard/kivy/<aplicacion>  
<aplicacion> debe ser un directorio con:
  - Archivo principal: main.py
  - Información sobre la app :android.txt

# Buildozer - preparación

Modificación del archivo buildozer.spec:

- source.dir = /path/del/proyecto/app
- version = 1.0
- requirements = kivy==master
- Path de android ndk(python-for-andorid)

```
android.ndk_path = ~/git/github/python-for-android/  
                    android/android-ndk-r8c
```

- Path de android sdk(python-for-andorid)

```
android.sdk_path = ~/git/github/python-for-android/  
                    android/android-sdk-linux
```

# Buildozer - Instalación

Una vez preparado el archivo

## Instalación de la app

- Copiar el archivo buildozer.spec al directorio del proyecto.
- buildozer -v android debug deploy

**Buildozer** compila, genera **apk** y la instala en el dispositivo conectado.

# Hello Pycon 2015

```
from kivy.app import App
from kivy.uix.button import Button
class TestApp(App):
    def build(self):
        return Button(text='Hello Pycon 2015')
TestApp().run()
```



# Armado

sub-classing la clase App

Implementar su método **build()**  
para que corra como una  
instancia de un Widget (es el root  
del árbol de widgets).

Instanciar esta clase y correr el  
método **run()**.

## Clase App

```
from kivy.app import App  
class TestApp(App):
```

## Método build()

```
def build(self):  
    return Button(text='Hello  
Pycon 2015')  
    — —
```

## Método run()

```
TestApp().run()  
— —
```

# Kv language

- Separa el diseño de la interfaz gráfica del código.
- Archivo separado con extensión **.kv**.
- Indentación identifica la estructura.
- Similar a css con html - glade con gtk.

## Ejemplo

```
<Hola>: # cada clase en la aplicación puede identificarse con
      una regla como esta
    Button: #se agrega el widget indicado.
        text:'Hola Pycon 2015' # configuración de una de las
                           propiedades del o layout
```

# Archivo py + kv

## main.py

```
from kivy.app import App

class TestApp(App):
    pass
if __name__=='__main__':
    TestApp().run()
```

## test.kv

*class TestApp(App) :⇒ test.kv*

```
Button:
    text:'Hola Pycon 2015'
```

Este archivo define el Widget Root que se conectará con la App y será la base del árbol widget de la aplicación.

# Tipos de Keywords

Hay 3 tipos de palabras claves específicas en kv:

- **app**: se refiere a la instancia de la aplicación
- **root**: es el widget base.
- **self**: se refiere al widget actual.

## test.kv

```
Button:  
    id: saludo  
    text:'Hola Pycon 2015!!!'  
    on_press: root.cambiar()
```

## main.py

```
class Hola(FloatLayout):  
    ...  
    def cambiar(self):  
        self.ids['saludo'].text  
        ='Chau'
```

```
<Hola>:  
    BoxLayout:  
        orientation: 'vertical' if  
            self.width < self.height  
        else 'horizontal'
```

# Comunicación por id

## test.kv

```
Image:  
    id: personaje  
    source: 'imagenes/  
        piepagina.png'
```

## main.py

```
self.pers = self.ids.  
    personaje  
self.pers.pos = (550,50) if  
    self.pers.pos == self.  
    center else self.center
```

```
self.pers = self.ids['  
    personaje']
```

# Comunicación por property

Properties: objetos que se modifican a través del patrón observer.

```
class ClockLayout(BoxLayout):
    time_prop = ObjectProperty(None)
    ...
    def cambiar(self):
        self.time_prop.text = "demo"
        self.ids['saludo'].text='Chau'
        self.pers.pos = (550,50) if self.
            pers.pos == self.center else
                self.center
```

```
<ClockLayout>:
    time_prop: time
    texto: leyendo
    Label:
        id: time
    Button:
        text:'Cambiar'
        on_press: root.
            cambiar()
```

# Importar

## main.py

```
from x.y import z as name  
from os.path import isdir  
import numpy as np
```

## test.kv

```
#:import name x.y.z  
#:import isdir os.path.isdir  
#:import np numpy
```

# No repeat yourself

```
GridLayout:  
    cols: 2  
    __row_default_height: 10  
GridLayout:  
    __cols: 2  
    __  row_default_height: 10
```

```
<GridLayout>:  
    cols: 2  
    #spacing: 10  
    row_default_height: 10
```

```
<GridMia@GridLayout>:  
    cols: 2  
    #spacing: 10  
    row_default_height:  
    .....  
GridMia:  
    Label:  
    text: 'Usuario:'
```

## Ver errores en Android:

- adb
- Logger

```
Logger.info('info: iniciando')
Logger.exception('Info: NO se pudo iniciar')
```

# Cambiar de pantallas

ScreenManager permite el cambio de pantalla.

```
<Info>:  
    screen_manager:  
        screen_manager  
        screen_manager.transition  
            .direction = 'right'  
    ScreenManager:  
        id: screen_manager  
    Screen:  
        name: 'inicio'  
        id: inicio  
        .....  
    Button:  
        on_release: root.iniciar(  
            inicio, 'piso_1')
```

```
def iniciar(self,  
           actual_screen,  
           next_screen):  
    self.  
        list_of_prev_screens  
            .append(  
                actual_screen.name)  
    self.screen_manager.  
        current =  
            next_screen  
    Logger.info('datos:  
        cambio pantalla')
```

# Cambiar de pantallas

## ScreenManager cargar dinámicamente

```
def jugar(self):
    Builder.load_file('tabaco.kv')
    import tabaco

def on_leave(self):
    Builder.unload_file('tabaco.kv')
    .....
class MenuPerder(Screen):
    pass
```

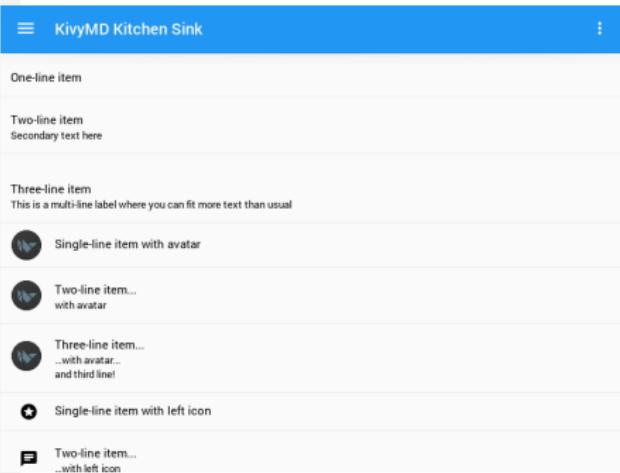
```
<MenuPerder>:
    id: 'perder'
    name: 'perder'
```

Toolbar:

```
id: toolbar
title: 'KivyMD Kitchen Sink'
background_color: app.theme_cls.
    primary_color
left_action_items: [['menu',
    lambda x: app.nav_drawer.
        toggle()]]
right_action_items: [['dots-
    vertical', lambda x: app.
        nav_drawer.toggle()]]
```

ScreenManager:

```
id: scc
Screen:
    name: bb
```



## Relacionar un widget con una acción desde kv y desde python.

```
Button:  
    text: "Cambiar"  
    on_press: root.cambiar("chau")
```

```
def cambiar(self, texto):  
    label = self.ids["hola"]  
    label.text = texto
```

Conectar desde python un botón con una función y pasarle parámetros

```
class Mota(Button):  
    -----  
    self.bind(on_release=  
        partial(self.  
            historial_mota,  
            historial))
```

```
<Mota>:  
    background_color: (0.0,  
                      0.0, 0.0, 0.0)  
    size_hint: (0.1, .1)  
    markup: True  
    valign: 'bottom'
```

# Creación de un servicio

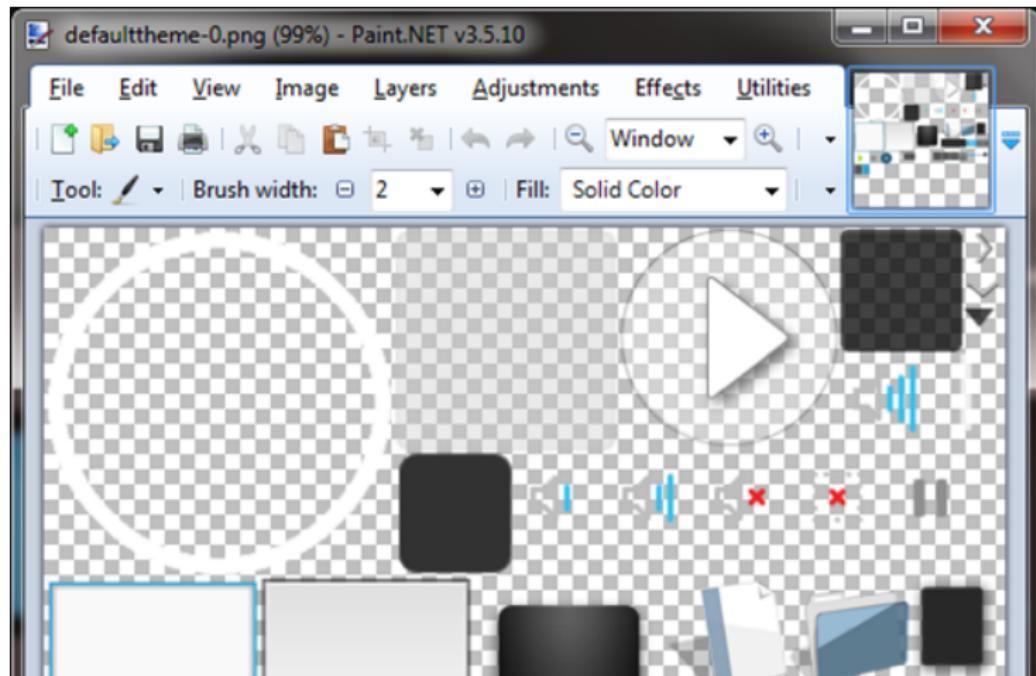
Recibir notificaciones cuando se suspende usando service.

Recuperar el estado al minimizar la aplicación.

```
def on_pause(self):
    if platform == 'android':
        from android import AndroidService
        service = AndroidService('Datos Sensores', 'running')
        service.start('service started')
        self.service = service
    return True
#~ #~
def on_resume(self):
    self.service.stop()
    self.service = None
--
from plyer import notification
def informar():
    notification.notify(app_name='Datos sensores - medición',
                        title=m['mota_id'], message='esta overheating - '+
                        str(m['temperatura'])+'C')
```

## Cambiar tema de los iconos

```
/usr/lib/python2.7/dist-packages/kivy/data/images/  
defaulttheme-0.png
```



```
Button:  
    id: conectar  
    text: 'Conectar'  
    background_color: (1.5,  
                      1.0, 0.0, 1.0)
```

## Cambiar color en los widget “Label” desde código Python.

```
def desdepypy(self):  
    label = self.ids["hola"]  
    ]  
    label.text = "[color  
    =13E400]"+ "chau" +"  
    [/color]"
```

```
Label:  
    id: hola  
    text: "hola"  
    markup: True
```

Introducción

oo  
oooo

Agregar color a label y botones

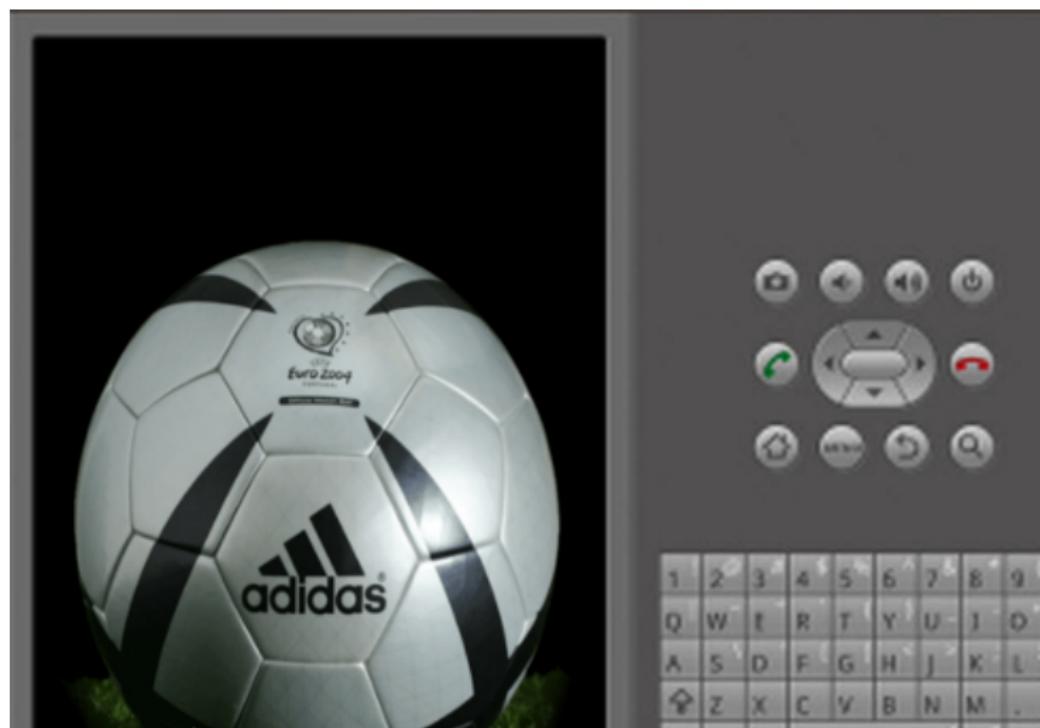
Entendiendo cómo se usa

oooo  
oooooooooooo

Diseño visual

oo●oooooooo

# Mi primera aplicación



Introducción

oo  
oooo

Agregar color a label y botones

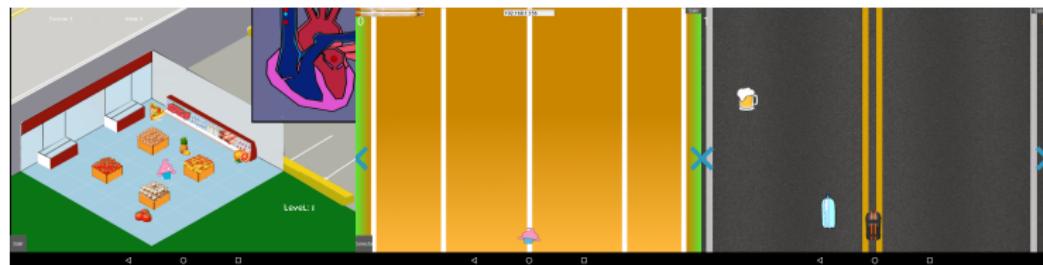
Entendiendo cómo se usa

oooo  
oooooooooooo

Diseño visual

oooo●oooooooo

# Nessie



Introducción

oo  
oooo

Agregar color a label y botones

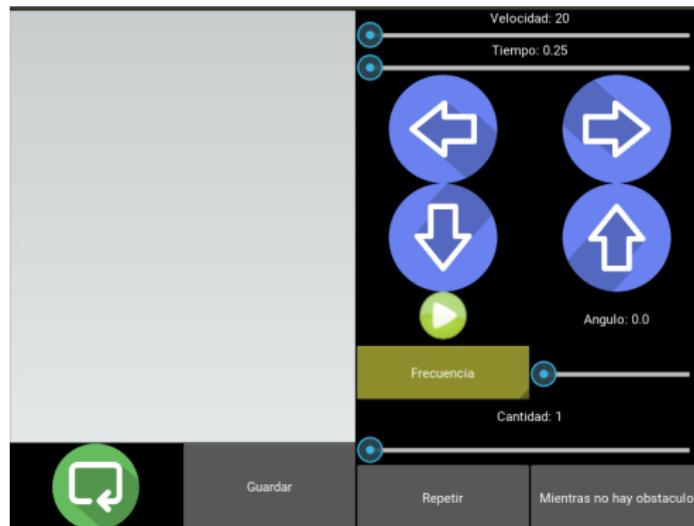
Entendiendo cómo se usa

oooo  
oooooooooooo

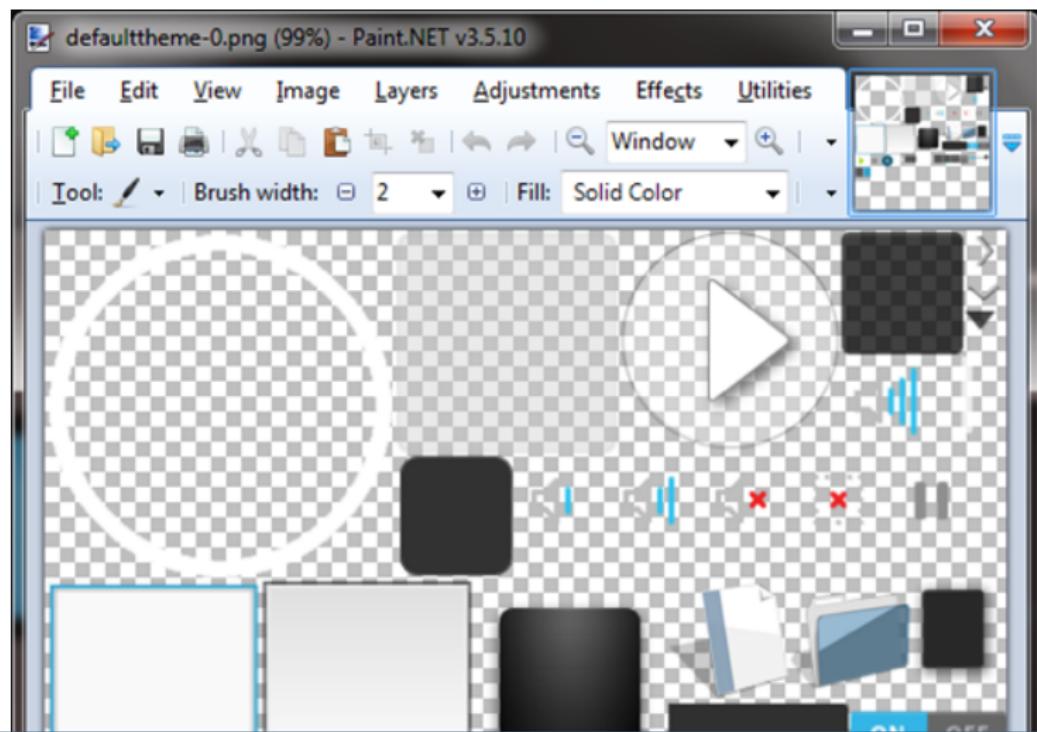
Diseño visual

oooo●oooooooo

# Robots



# Diseño visual

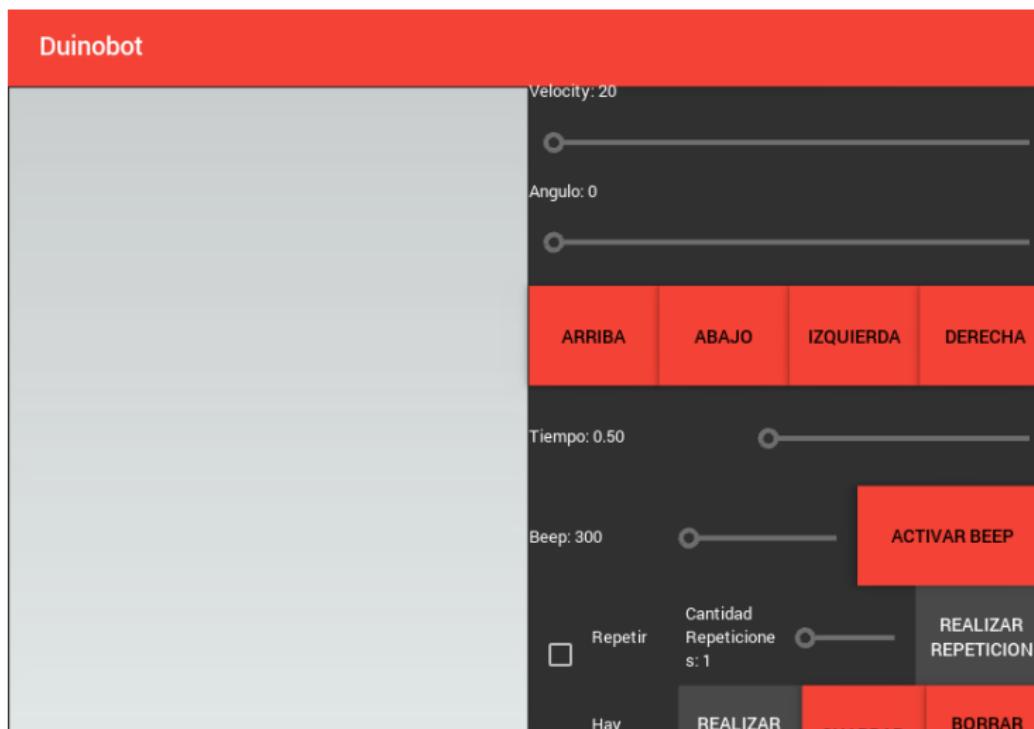


# Kivy con material design

The screenshot displays a mobile application interface titled "KivyMD Kitchen Sink". The top navigation bar is blue with the title "KivyMD Kitchen Sink" and a three-line menu icon on the left, and a vertical ellipsis icon on the right. Below the navigation bar, the screen is divided into several horizontal sections, each representing a different type of card item:

- One-line item:** A single-line text entry.
- Two-line item:** Two-line text entry with secondary text below it.
- Three-line item:** Three-line text entry with a descriptive subtitle below it.
- Single-line item with avatar:** Single-line text entry next to a circular user icon.
- Two-line item... with avatar:** Two-line text entry next to a circular user icon.
- Three-line item... with avatar... and third line!** Three-line text entry next to a circular user icon, with an ellipsis between the first and second lines.
- Single-line item with left icon:** Single-line text entry next to a star-shaped icon.

# Aplicación con material design



# Módulos - plataformas para aprovechar

- Plyer
- Pyjnius

preguntas?

- @entrerrianas
- Sofía Martin y Lucas Chiacchio