

Introducción al Desarrollo de Aplicaciones en Android

- Ing. Skrauba Axel

Framework

¿Qué es?

Un **entorno de trabajo** o **marco de trabajo** es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

¿Ventajas?

Utilizar un *framework* permite **agilizar los procesos** de desarrollo ya que evita tener que escribir código de forma repetitiva, asegura unas **buenas prácticas** y la **consistencia** del código.

Kivy

The Open Source Python App Development Framework.

Build and distribute
beautiful Python cross-
platform GUI apps with
ease.

¿Ventajas?

Kivy ha sido creado para ser fácil de usar, multiplataforma y rápido.

Con un solo código base, podrá implementar aplicaciones en Windows, Linux, macOS, iOS y Android.

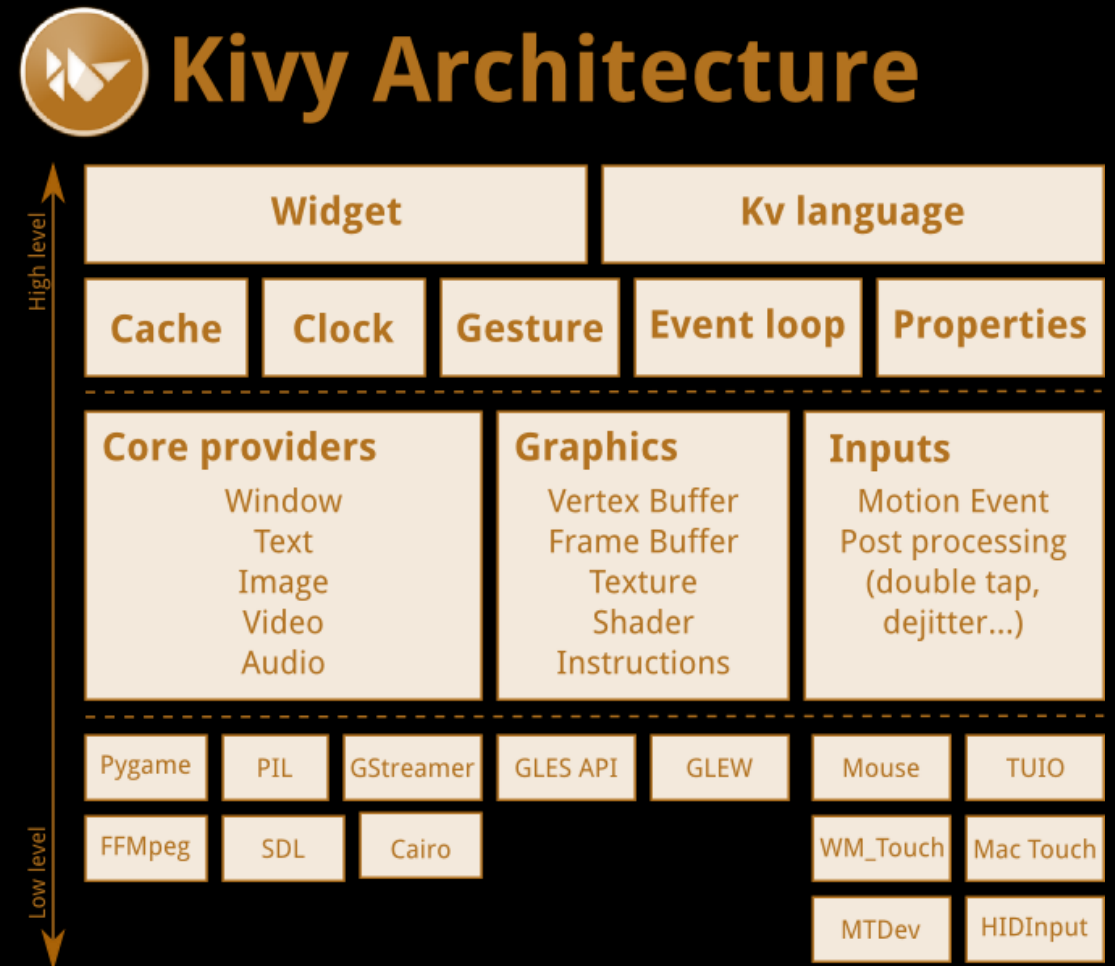


<https://kivy.org/>

Kivy: arquitectura

Diseño de *Kivy* desde el punto de vista de la ingeniería de software.

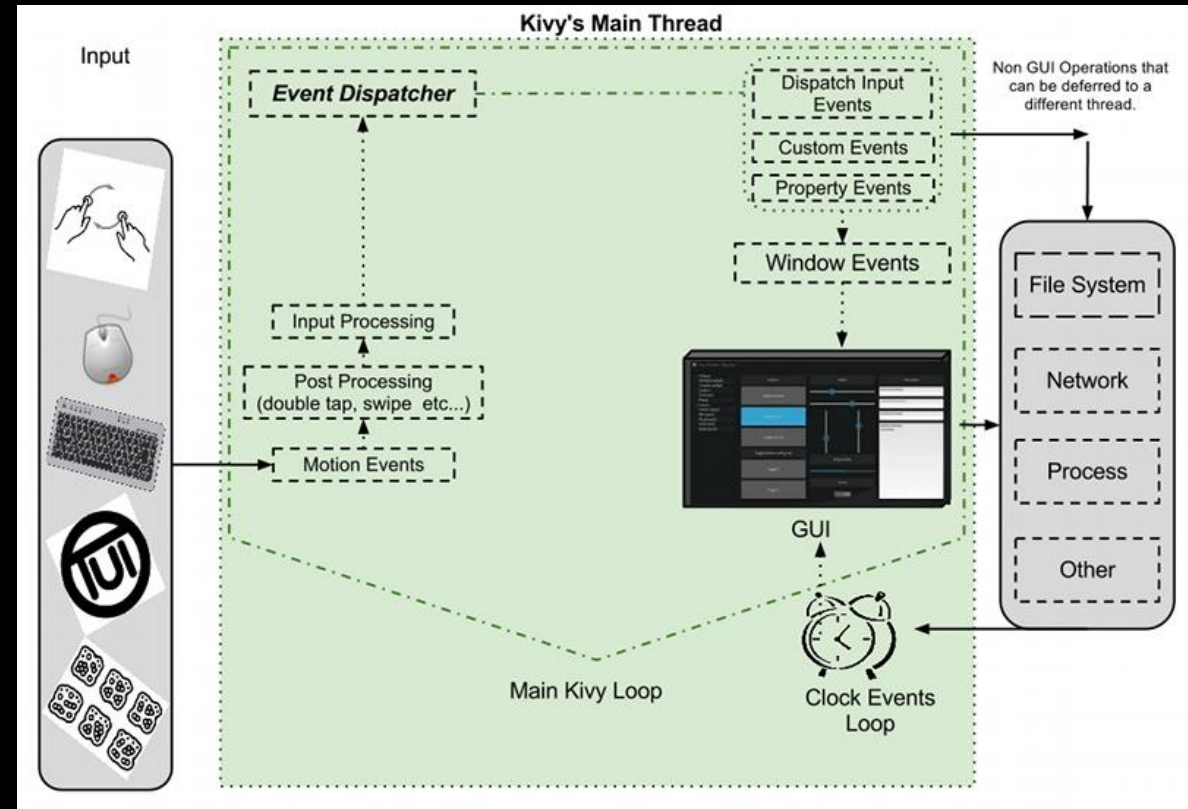
<https://kivy.org/doc/stable/guide/architecture.html>



Kivy: gestión de eventos

Los eventos son una parte importante de la programación de Kivy. Eso puede no ser sorprendente para aquellos con experiencia en desarrollo de GUI, pero es un concepto importante para los recién llegados.

<https://kivy.org/doc/stable/guide/events.html>



Primer Aplicación



```
1  from kivy.app import App
2  from kivy.uix.label import Label
3
4
5  class MyApp(App):
6
7      def build(self):
8          return Label(text='Hello world')
9
10
11  if __name__ == '__main__':
12      MyApp().run()
```

Herencia

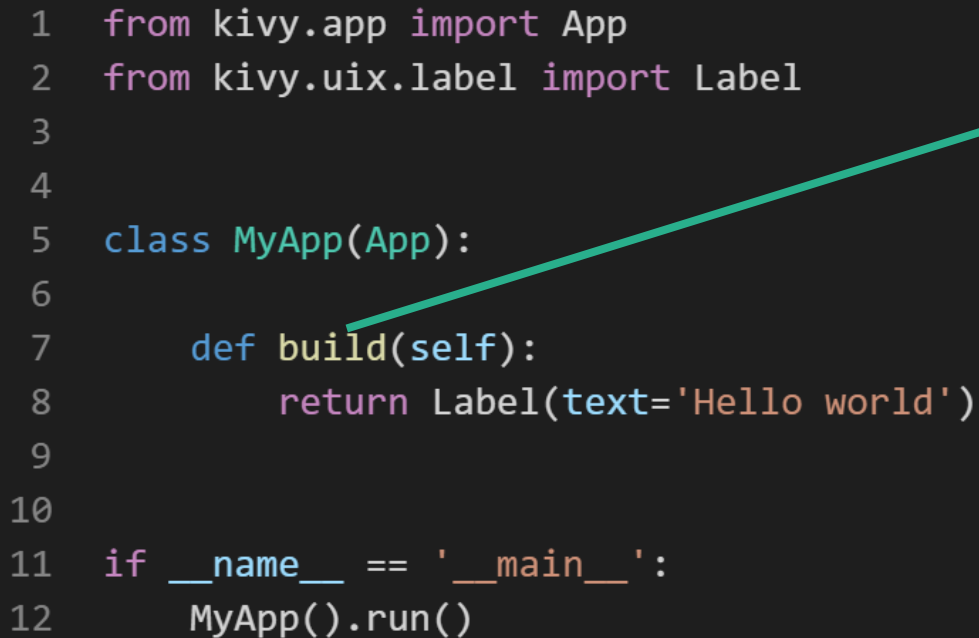
La clase *App* es la base para crear aplicaciones Kivy.

Widget

Método

Instancia de la clase de aplicación específica y luego, se inicia el ciclo de vida de la aplicación con el método *App.run()* de la instancia.

Primer Aplicación



```
1  from kivy.app import App
2  from kivy.uix.label import Label
3
4
5  class MyApp(App):
6
7      def build(self):
8          return Label(text='Hello world')
9
10
11  if __name__ == '__main__':
12      MyApp().run()
```

Para inicializar la aplicación con un **árbol de widgets**, se sobrescribe el método *build()* en la clase de aplicación y se devuelve el árbol de *widgets* construido.

Segunda Aplicación

```
1 from kivy.app import App
2 from kivy.uix.boxlayout import BoxLayout
3 from kivy.uix.button import Button
4 from kivy.uix.textinput import TextInput
5
6 class MiApp(App):
7
8     def build(self):
9         self.title = "Mi Aplicación"
10        b = BoxLayout(orientation='vertical')
11
12        self.txt_input = TextInput(multiline=False)
13        b.add_widget(self.txt_input)
14
15        btn = Button(text="Saludar",
16                    on_press=self.saludar)
17        b.add_widget(btn)
18
19        return b
20
21    def saludar(self, obj):
22        nombre = self.txt_input.text
23        print("¡Hola {}!".format(nombre))
24        self.txt_input.text = ""
25
26 if __name__ == "__main__":
27     MiApp().run()
```

Para inicializar la aplicación con un **árbol de widgets**... Una especie de “caja” para incorporar varios *widgets*.

Evento: al presionar... lanza la ejecución de una porción de código.

<https://kivy.org/doc/stable/api-kivy.uix.html>

Segunda Aplicación

```
1 from kivy.app import App
2 from kivy.uix.boxlayout import BoxLayout
3 from kivy.uix.button import Button
4 from kivy.uix.textinput import TextInput
5
6 class MiApp(App):
7
8     def build(self):
9         self.title = "Mi Aplicación"
10        b = BoxLayout(orientation='vertical')
11
12        self.txt_input = TextInput(multiline=False)
13        b.add_widget(self.txt_input)
14
15        btn = Button(text="Saludar",
16                    on_press=self.saludar)
17        b.add_widget(btn)
18
19        return b
20
21    def saludar(self, obj):
22        nombre = self.txt_input.text
23        print("¡Hola {}!".format(nombre))
24        self.txt_input.text = ""
25
26 if __name__ == "__main__":
27     MiApp().run()
```

La “caja” para ubicar las “otras cosas”.

Nuevo *widget*

Agregado a la “caja”.

Se devuelve, otra vez, lo que hay que renderizar. Ahora hay varias cosas.

<https://kivy.org/doc/stable/api-kivy.uix.html>

Tercer Aplicación

```
1 from kivy.app import App
2 from kivy.uix.boxlayout import BoxLayout
3 from kivy.uix.button import Button
4 from kivy.uix.label import Label
5
6 class MiApp(App):
7
8     def build(self):
9         layout = BoxLayout(orientation='horizontal')
10
11         btn = Button(text='Presioname')
12         btn.bind(on_press=self.presionado)
13         btn.bind(on_release=self.liberado)
14
15         self.label = Label(text='Aún no presionado')
16         layout.add_widget(self.label)
17         layout.add_widget(btn)
18
19         return layout
20
21     def presionado(self, instance):
22         self.label.text = "¡Presionado!"
23
24     def liberado(self, instance):
25         self.label.text = "¡Liberado!"
26
27 MiApp().run()
```

Otra forma de *setear* parámetros

2 eventos diferentes

El segundo parámetros es el objeto desde el que se invoca al método.

<https://kivy.org/doc/stable/api-kivy.uix.html>

Cuarta Aplicación

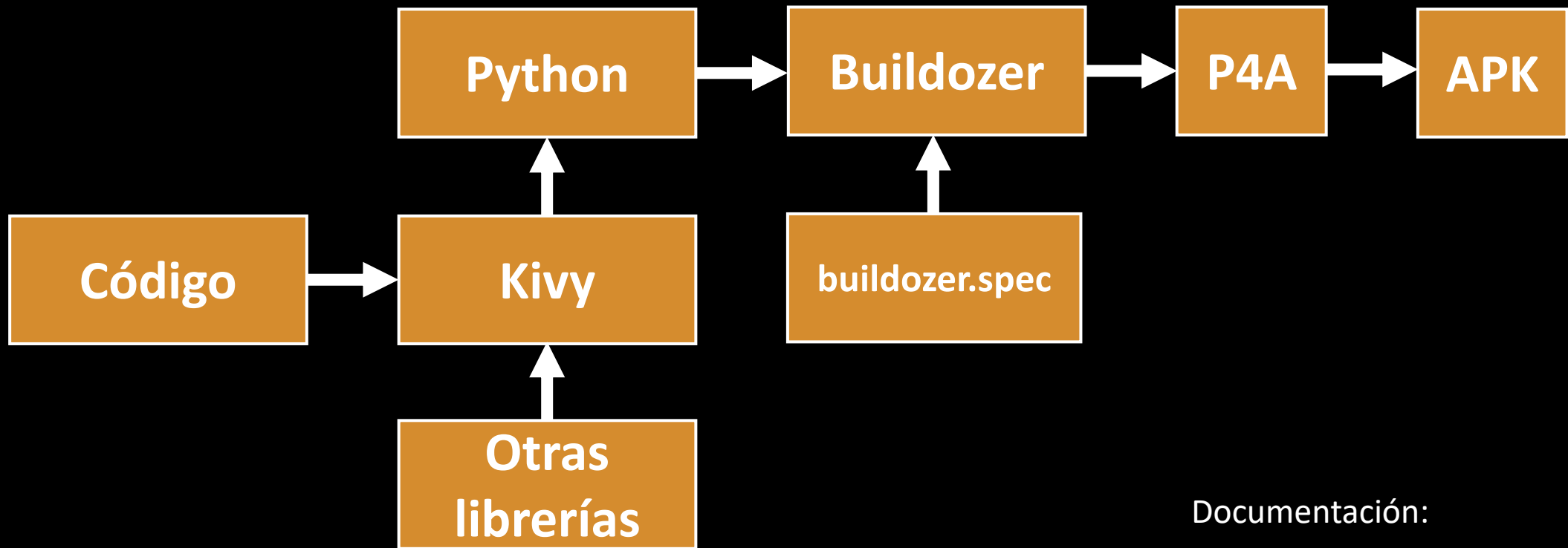
```
1 from kivy.app import App
2 from kivy.ui.boxlayout import BoxLayout
3 from kivy.ui.textinput import TextInput
4 from kivy.ui.button import Button
5 from kivy.ui.checkbox import CheckBox
6
7 class MiApp(App):
8
9     def build(self):
10         layout = BoxLayout(orientation='vertical')
11
12         self.pwd1 = TextInput(password=True)
13         layout.add_widget(self.pwd1)
14
15         self.pwd2 = TextInput(password=True)
16         layout.add_widget(self.pwd2)
17
18         self.chk = CheckBox()
19         self.chk.bind(active=self.on_checkbox_active)
20         layout.add_widget(self.chk)
21
22         btn = Button(text='Verificar', on_press=self.verificar)
23         layout.add_widget(btn)
24
25         return layout
26
27     def verificar(self, instance):
28         if self.pwd1.text == self.pwd2.text:
29             print("Las contraseñas coinciden")
30         else:
31             print("Las contraseñas no coinciden")
32
33     def on_checkbox_active(self, checkbox, value):
34         if value:
35             self.pwd1.password = False
36             self.pwd2.password = False
37         else:
38             self.pwd1.password = True
39             self.pwd2.password = True
40
41 MiApp().run()
```

Para configurar el *TextInput* como contraseña se utiliza la propiedad *password=True*.

Ciertos *widgets* devuelven estados. El *CheckBox*, un booleano.

Este método cambia la propiedad *password* de los *TextInput* según el estado del *checkbox*

Kivy: ¿Cómo obtengo un *.apk*?



Documentación:

- <https://kivy.org/doc/stable/>
- <https://github.com/kivy/buildozer>

Preguntas...

- Dudas
- Sugerencias
- Cuestiones