KIVY (PYTHON)

Carlos Maldonado

5IV7

Introducción

Kivy es un *framework* de código abierto, escrito en Python, diseñado para el desarrollo rápido de aplicaciones con interfaces de usuario innovadoras, especialmente aquellas que soportan interacción **multitáctil** (NUI - Natural User Interface). Su principal atractivo radica en la capacidad de crear una aplicación con un **único código base** que funciona en una amplia gama de dispositivos y sistemas operativos.

¿Qué es Kivy?

Técnicamente, Kivy es una **biblioteca de Python** de código abierto para el desarrollo de *software* de aplicación multiplataforma.

- **Framework para Python:** Permite a los desarrolladores de Python crear interfaces gráficas de usuario (GUI) para diversas plataformas.
- **Multiplataforma:** La misma aplicación puede ejecutarse en sistemas de escritorio (**Windows, Linux, macOS**) y en dispositivos móviles (**Android, iOS**).
- **NUI (Natural User Interface):** Está construido desde cero para adaptarse a las necesidades de las aplicaciones táctiles y multitáctiles.

Historia y evolución de Kivy

Kivy surgió de la necesidad de un *framework* que permitiera a los desarrolladores de Python centrarse en la creación de aplicaciones personalizadas y altamente interactivas, aprovechando la naturaleza dinámica del lenguaje Python.

- **Filosofía:** Se concibió con el objetivo de facilitar el desarrollo rápido de prototipos y aplicaciones, especialmente en el ámbito de las interfaces multitáctiles.
- **Evolución:** Kivy ha sido desarrollado, respaldado y utilizado profesionalmente por una comunidad de código abierto.
    - En **2012**, recibió una subvención de la Python Software Foundation (PSF) para su portabilidad a Python 3.3, lo que marcó un hito en su madurez.
    - Ha participado en varios programas **Google Summer of Code (GSoC)**, lo que demuestra su constante desarrollo y evolución.
    - Proyectos como **KivyMD** han surgido para proporcionar un *look and feel* basado en el diseño *Material Design* de Google, mejorando su atractivo visual en el entorno de Android.

Características principales

Las características clave de Kivy lo hacen una herramienta potente y flexible:

Característica	Descripción
Multiplataforma	Un solo código fuente funciona en Windows, Linux, macOS, Android e iOS.
Código Abierto y Licencia MIT	Es completamente gratuito, incluso para uso comercial, lo que promueve su adopción y desarrollo.
Motor Gráfico Propio	Está construido sobre **OpenGL ES 2**, usando un *pipeline* de gráficos moderno y rápido. Esto garantiza que la aplicación se vea **exactamente igual** en todos los dispositivos, ya que no utiliza *widgets* nativos del sistema operativo.
Lenguaje KV	Un lenguaje declarativo propio que permite definir la interfaz de usuario (*widgets* y *layouts*) separándola de la lógica de la aplicación (escrita en Python), facilitando el prototipado rápido y el mantenimiento.
Soporte Multitáctil	Ofrece soporte nativo y extenso para eventos de entrada, incluyendo **multitouch** (TUIO), teclado, y ratón.
Rendimiento	Las partes críticas (principalmente el motor gráfico) están implementadas en **C** y **Cython**, garantizando un alto rendimiento a pesar de estar escrito en Python.

Arquitectura de Kivy

La arquitectura de Kivy se basa en la separación de la interfaz de usuario y la lógica de la aplicación, utilizando varios componentes clave:

- **App:** Es la clase base de toda aplicación Kivy, encargada de inicializar y ejecutar el ciclo de vida de la aplicación.
- **Widget:** El bloque de construcción fundamental de la interfaz de usuario (botones, etiquetas, cajas de texto, etc.). Proporciona un lienzo para dibujar y recibe eventos de entrada.

- **Layout:** Son los contenedores que se utilizan para organizar y posicionar los *widgets* de forma adecuada en la pantalla (como BoxLayout, GridLayout, FloatLayout, etc.).
- **Lenguaje KV (Kvlang):** El lenguaje declarativo que se utiliza para describir la jerarquía de *widgets* y sus propiedades. Esto se carga y se enlaza automáticamente a la lógica de Python.
- **Motor Gráfico (OpenGL ES 2):** Es el responsable de dibujar todos los *widgets* desde cero, asegurando una apariencia uniforme en todas las plataformas.
- **Propiedades y Eventos:** Kivy utiliza un sistema de propiedades (similares a los atributos de Python) que permiten enlazar y reaccionar a los cambios en la interfaz de manera eficiente (Manejo de Eventos).

Instalación y configuración del entorno

La instalación de Kivy es sencilla gracias a su disponibilidad a través del gestor de paquetes de Python, **pip**.

Requisitos previos:

1. Tener **Python** (versión 3.x recomendada) instalado.
2. Tener el gestor de paquetes **pip** actualizado.

Comando de Instalación (Recomendado):

Abre tu terminal o símbolo del sistema y ejecuta:

pip install kivy

Comandos y estructuras básicas

Estructura Mínima de una Aplicación Kivy

Toda aplicación Kivy debe heredar de la clase kivy.app.App e implementar el método build():

```
from kivy.app import App
from kivy.uix.label import Label
class MiAplicacion(App):
  def build(self=\_):
    return Label(text='Mi Primera App Kivy')
if __name__ == '__main__':
  MiAplicacion().run()
```

El Lenguaje KV

El lenguaje KV se usa para definir la interfaz en un archivo separado, llamado automáticamente si tiene el mismo nombre que la clase de la aplicación (en minúsculas y sin "App"), por ejemplo: miapplicacion.kv.

```
BoxLayout:
  orientation: 'vertical'
  Button:
    text: 'Presioname'
  Label:
    text: 'Etiqueta en KV'
```

Widgets y Layouts

Widgets Básicos

Un *widget* es cualquier elemento de la interfaz de usuario:

Widget	**Descripción**
Label	Muestra texto estático
Button	Un botón interactivo
TextInput	Campo para la entrada de texto por parte del usuario
Image	Muestra una imagen
Slider	Permite seleccionar un valor dentro de un rango

Layouts (Contenedores)

Los *layouts* organizan los *widgets* hijos en la pantalla:

Layout	**Descripción**
BoxLayout	Organiza los *widgets* en una caja horizontal o vertical.
GridLayout	Organiza los *widgets* en una cuadrícula (filas y columnas).
FloatLayout	Permite posicionar los *widgets* con coordenadas relativas o absolutas.
RelativeLayout	Posiciona los *widgets* en relación con el tamaño y la posición del propio *layout*.
StackLayout	Organiza los *widgets* apilándolos según el tamaño disponible.

Manejo de eventos y propiedades

Propiedades

Los *widgets* tienen **propiedades** (Property) que definen su comportamiento y apariencia (ej. text, size, pos, color). Kivy permite enlazar la lógica de Python a los cambios en estas propiedades.

Manejo de Eventos

La interacción del usuario se gestiona mediante eventos. El evento más común es el de **presionar un botón** (on_press).

Ejemplo de manejo de evento en Python:

```
from kivy.uix.button import Button

def al_presionar(instancia):
    print("¡El botón ha sido presionado!")
    instancia.text = "¡Presionado!"

boton = Button(text='Haz clic')
boton.bind(on_press=al_presionar)
```

Ejemplo de manejo de evento en KV:

```
Button:
    text: 'Haz clic'
    # Llama al método de la clase App (o del root widget)
    on_press: root.mi_metodo_de_accion()
```

Ejemplo 1: Aplicación básica "Hola Mundo"

Este ejemplo combina la clase App con el lenguaje KV para una estructura limpia.

Archivo holamundo.py:

```
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout # Importamos el layout

class HolaMundoApp(App):
    def build(self):
        return BoxLayout()

if __name__ == '__main__':
    HolaMundoApp().run()
```

Archivo holamundo.kv:

```
BoxLayout:
  orientation: 'vertical' # Organización vertical
  spacing: 10 # Espaciado entre elementos

  Label:
    text: '¡Hola, Mundo desde Kivy!'
    font_size: '30sp'

  Button:
    text: 'Cerrar Aplicación'
    size_hint_y: 0.2 # Ocupa el 20% del alto disponible
    on_release: app.stop() # Detiene la aplicación al soltar el botón
```

Ejemplo 2: Calculadora con Kivy

Una estructura de calculadora utiliza típicamente un GridLayout para los botones y un TextInput o Label para la pantalla.

Estructura del Archivo .kv (fragmento):

```
BoxLayout: # Layout principal vertical
  orientation: 'vertical'
  padding: 10
  TextInput:
    id: calc_input # ID para acceder desde Python
    text: '0'
    font_size: 50
    size_hint_y: 0.3
  GridLayout:
    cols: 4
    spacing: 5
    size_hint_y: 0.7 # Ocupa el 70% restante
    Button:
      text: '7'
      on_press: app.agregar_numero('7')
    Button:
      text: '8'
      on_press: app.agregar_numero('8')
```

Estructura del Archivo .py (fragmento):

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

  def agregar_numero(self, numero):
    input_widget = self.root.ids.calc_input

    if input_widget.text == '0':
      input_widget.text = numero
    else:
      input_widget.text += numero

if __name__ == '__main__':
  CalculadoraApp().run()
```

Ejecución de aplicaciones Kivy

Para ejecutar cualquier aplicación Kivy, simplemente navega hasta el directorio que contiene el archivo .py (y el .kv si lo usas) y ejecuta el *script* de Python:

```
python nombre_del_archivo.py
```

Ventajas y desventajas

Ventajas de Kivy

Ventaja	Descripción
Desarrollo Rápido	El uso de Python y el lenguaje KV permite prototipar y desarrollar rápidamente.
Verdadera Multiplataforma	Un código funciona en 5 sistemas operativos principales (Win, Linux, macOS, Android, iOS).
Apariencia Consistente	Al dibujar sus propios *widgets* con OpenGL, la interfaz es idéntica en todas las plataformas.
Python	Acceso a miles de librerías de Python para funcionalidades complejas (redes, ciencia de datos, etc.).
Licencia Amigable	Licencia MIT, libre de usar para proyectos comerciales.

Desventajas de Kivy

Desventaja	Descripción
No Nativo (UI)	Al no usar *widgets* nativos, las aplicaciones Kivy no tienen la apariencia y sensación nativa de cada SO (aunque KivyMD ayuda a mitigar esto en Android).
Embalaje (Packaging) Complejo	El proceso para generar binarios o paquetes de apps (APK, IPA) es más complejo que para otros *frameworks* de *cross-platform* como Flutter o React Native.
Curva de Aprendizaje	Aprender el Lenguaje KV, la arquitectura de Kivy y las herramientas de *packaging* puede llevar tiempo.
Ecosistema/Popularidad	Es menos popular que alternativas como Flutter o React Native, lo que se traduce en una comunidad más pequeña y puede hacer más difícil encontrar desarrolladores con experiencia.

Usos y aplicaciones en la industria

Kivy es ideal para proyectos donde la **interfaz de usuario natural (NUI)** y la **rapidez de desarrollo** son claves:

- **Prototipos y MVPs:** Su velocidad con Python lo hace perfecto para probar ideas rápidamente.
- **Aplicaciones Científicas/Educativas:** Permite crear interfaces ricas e interactivas para simulación o visualización de datos.
- **Aplicaciones de Punto de Venta (POS) y Kioscos:** Es robusto para entornos de pantalla táctil fija.
- **Juegos y Gráficos 2D Simples:** El motor OpenGL ES 2 lo hace apto para gráficos y pequeñas animaciones.
- **Automatización y IoT:** Como interfaz para sistemas de control basados en Python.

Conclusiones

Kivy es una solución robusta y única que permite a los desarrolladores de Python entrar en el mundo de las aplicaciones multiplataforma, incluyendo el desarrollo móvil. Destaca por su enfoque en la interfaz natural y su capacidad de mantener una apariencia consistente en todos los sistemas operativos.

Es la opción ideal si:

1. Tu *backend* o lógica de negocio ya está fuertemente atada a **Python**.
2. Buscas una **apariencia uniforme** en todas las plataformas.
3. Necesitas capacidades **multitáctiles** avanzadas.

Sin embargo, para proyectos donde una sensación 100% nativa o una adopción masiva es el objetivo principal, podrías considerar alternativas más populares.

Bibliografía

1. **Kivy Project.** (2024). *Documentación oficial de Kivy.* Recuperado de: https://kivy.org/doc/stable/
2. **Python Software Foundation.** (2024). *Documentación oficial de Python*. Disponible en: https://www.python.org/
3. **González, M.** (2023). *Desarrollo de aplicaciones multiplataforma con Python y Kivy.* Editorial Alfaomega.
4. **Real Python.** (2024). *Tutoriales de desarrollo con Kivy.* Disponible en: https://realpython.com/
5. **Stack Overflow en Español.** (2024). *Sección de preguntas sobre Kivy*. Disponible en: https://es.stackoverflow.com/questions/tagged/kivy
6. **López, D.** (2022). *Interfaces gráficas con Python y Kivy.* Universidad Nacional Autónoma de México (UNAM).
7. **Martínez, A.** (2023). *Programación con Python: de la lógica al desarrollo de apps móviles*. Editorial Ra-Ma.
8. **KivyMD Team.** (2024). *KivyMD Documentation*. Recuperado de: https://kivymd.readthedocs.io/
9. **Geeks for Geeks.** (2024). *Introducción a Kivy en Python.* Disponible en: https://www.geeksforgeeks.org/python-kivy/
10. **Smith, J.** (2023). *Cross-platform GUI Development using Kivy and Python*. O'Reilly Media.