Análisis de 3 bibliotecas GUI para Python

Elian Toapanta, Iván Zambrano y Marlon Torres

Resumen - En este documento se presentan tres bibliotecas o librerías GUI (graphical user interface), el cual es un programa informático que presenta al usuario una interfaz gráfica, utilizando imágenes y objetos gráficos para poder presentar información y las acciones que se requieran en el sistema. Estas tres librerías son del lenguaje de programación Python, que con el mismo se realizo una calculadora la cual puede realizar las 4 operaciones básicas.

Índice de Términos – Python, GUI, bibliotecas.

I. INTRODUCCIÓN

El desarrollo es una parte fundamental y no hay escasez de lenguajes de programación, siendo Python la tendencia. Python es un lenguaje de programación interactivo y comenzar a programar un marco GUI (interfaz gráfica de usuario) no es una tarea difícil. Python tiene una amplia gama de opciones para marcos de GUI. Desde marcos multiplataforma a marcos específicos de plataforma, el wiki de Python los enumera a todos.

II. PYTHON

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. (Python, n.d.). Python fue creado a finales de los ochenta por Guido van Rossum en el Centro para las Matemáticas y la Informática (CWI, Centrum Wiskunde & Informatica), en los Países Bajos, como un sucesor del lenguaje de programación ABC, capaz de manejar excepciones e interactuar con el sistema operativo Amoeba. El nombre del lenguaje proviene de la afición de un grupo británico de comediantes conocidos como Monty Python (Python, n.d.)

III. GUI

Una GUI es una interfaz de usuario gráfica (en lugar de puramente textual) para una computadora. Al leer cualquier página web o aplicación estás viendo e interactuando con una GUI o la interfaz gráfica de usuario. El término entró en existencia porque las primeras interfaces de usuario interactivas a las computadoras no eran gráficas; estaban orientadas a texto y teclado, y por lo general consistían en comandos que tenía que recordar y las respuestas de la computadora que eran infamemente breves.

IV. LIBRERÍAS GUI PARA PYTHON

Para un profesional de web, el desarrollo es una parte fundamental y en temas de lenguajes de programación hay varios, pero siendo Python la tendencia. Python es un lenguaje de programación interactivo y comenzar a programar un marco GUI (interfaz gráfica de usuario) no es una tarea difícil. Python tiene una amplia gama de opciones para marcos de GUI. Desde marcos multiplataforma a marcos específicos de plataforma (Fatima, 2017).

A. PyQt

PyQt implementa la popular biblioteca Qt (es un framework multiplataforma orientado a objetos ampliamente usado para desarrollar programas que utilicen interfaz gráfica de usuario que utiliza el lenguaje C++), por lo que, si está familiarizado con el desarrollo de Qt en otro idioma, quizás desde el desarrollo de aplicaciones nativas para KDE(es una comunidad internacional que desarrolla software libre.) u otro entorno de escritorio con Qt, es posibles que ya esté familiarizado con Qt. Esto abre la posibilidad de desarrollar aplicaciones en Python que tengan un aspecto familiar en muchas plataformas, al tiempo que aprovechan las herramientas y el conocimiento de la gran comunidad de Qt. (Chazallet, 2015)

```
import sys
from PyQt5.QtWidgets import QMainWindow, QApplication, QLabel
class VentanaPrincipal(QMainWindow):
    def __init__(self):
        QMainWindow.__init__(self)
        self.setFixedSize(500, 500)
        self.setWindowTitle("Hola mundo")
app = QApplication(sys.argv)
ventanita = VentanaPrincipal()
ventanita.show()
sys.exit(app.exec())
```

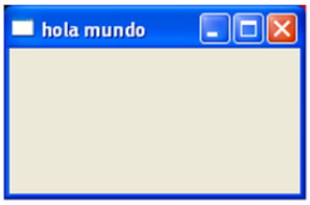

Fig. 1. Ejemplo de la aplicación de la librería PyQt

.

B. *WxPython*

WxPython le da a un desarrollador una forma de beneficiarse de una biblioteca de GUI multiplataforma, con una licencia clara, al mismo tiempo que brinda los beneficios de Python. Al igual que wxWidgets y Python, wxPython es gratuito y de código abierto, y está disponible para su uso y distribución en proyectos tanto gratuitos como comerciales sin el requisito de distribuir su código fuente (Chazallet, 2015).

Wxpython - Empezando Con Wxpython | Wxpython Tutorial, n.d. dice que: Suite GUI completa que incluye (pero no se limita a): Windows (incluyendo Windows MDI), Magos, Marcos y MiniFrames, diálogos, estándar, avanzado y personalizado, libros, árboles, cuadrículas y controles de vista de datos, tiene indicadores, controles deslizantes, giradores, animaciones, portapapeles, arrastrar y soltar, también tiene compatibilidad con HTML, PDF y visor de imágenes.

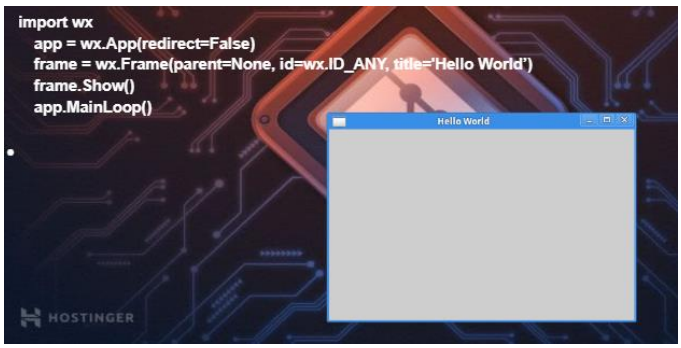

Fig. 2. Ejemplo de la aplicación de la librería WxPython

C. *Tkinter*

Si hubiera un único paquete que podría denominarse el kit de herramientas GUI “estándar” para Python, sería Tkinter. Tkinter es una envoltura alrededor de Tcl/Tk, una interfaz gráfica popular y el emparejamiento de idiomas que se popularizó por primera vez a principios de los 90. La ventaja de elegir Tkinter es la gran cantidad de recursos, incluidos libros y ejemplos de código, así como una gran comunidad de usuarios que pueden ayudarlo si tiene alguna pregunta. Los ejemplos simples son fáciles de comenzar y bastante legibles para los humanos. Tkinter está disponible bajo la licencia de Python, además de la licencia BSD de Tcl/Tk. (Chazallet, 2015)

Fig. 3. Ejemplo de la aplicación de la librería Tkinter

V. IMPLEMENTAR UNA CALCULADORA BÁSICA (SUMA, RESTA MULTIPLICACIÓN, DIVISIÓN).

```
1
2 from tkinter import Tk,Text,Button,END,re
3
4 class Interfaz:
5     def __init__(self, ventana):
6         #Inicializar la ventana con un título
7         self.ventana=ventana
8         self.ventana.title("Calculadora Grupo 4")
9
10         #Agregar una caja de texto para que sea la pantalla de la calculadora
11         self.pantalla=Text(self.ventana, state="disabled", width=40, height=3, backgrou
nd="orange", foreground="white", font=("Helvetica",15))
12
13         #Ubicar la pantalla en la ventana
14         self.pantalla.grid(row=0, column=0, columnspan=4, padx=5, pady=5)
15
16         #Inicializar la operación mostrada en pantalla como string vacío
17         self.operacion=""
```

Fig. 4. Primera parte del codigo de la CArculadora en Python

```
#Crear los botones de la calculadora
boton1=self.crearBoton(7)
boton2=self.crearBoton(8)
boton3=self.crearBoton(9)
boton4=self.crearBoton(u"\u232B",escribir=False)
boton5=self.crearBoton(4)
boton6=self.crearBoton(5)
boton7=self.crearBoton(6)
boton8=self.crearBoton(u"\u00F7")
boton9=self.crearBoton(1)
boton10=self.crearBoton(2)
boton11=self.crearBoton(3)
boton12=self.crearBoton("*")
boton13=self.crearBoton(".")
boton14=self.crearBoton(0)
boton15=self.crearBoton("+")
boton16=self.crearBoton("-")
boton17=self.crearBoton("=",escribir=False,ancho=20,alto=2)
```

Fig. 5. Segunda parte del codigo de la CArculadora en Python

```
38         #Ubicar los botones con el gestor grid
39         botones=[boton1, boton2, boton3, boton4, boton5, boton6, boton7, boton8, boton9
, boton10, boton11, boton12, boton13, boton14, boton15, boton16, boton17]
40         contador=0
41         for fila in range(1,5):
42             for columna in range(4):
43                 botones[contador].grid(row=fila,column=columna)
44                 contador+=1
45         #Ubicar el último botón al final
46         botones[16].grid(row=5,column=0,columnspan=4)
47
48         return
49
50
51     #Crea un botón mostrando el valor pasado por parámetro
52     def crearBoton(self, valor, escribir=True, ancho=9, alto=1):
53         return Button(self.ventana, text=valor, width=ancho, height=alto, font=("Helvet
ica",15), command=lambda:self.click(valor,escribir))
54
```

Fig. 6. Tercera parte del codigo de la CArculadora en Python

```
    #Controla el evento disparado al hacer click en un botón
    def click(self, texto, escribir):
      #Si el parámetro 'escribir' es True, entonces el parámetro texto debe mostrarse e
n pantalla. Si es False, no.
        if not escribir:
            #Sólo calcular si hay una operación a ser evaluada y si el usuario presionó
 '='
            if texto=="=" and self.operacion!="":
                #Reemplazar el valor unicode de la división por el operador división de
 Python '/'
                self.operacion=re.sub(u"\u00F7", "/", self.operacion)#re=remplaza el si
mbolo
                resultado=str(eval(self.operacion))#eval=evaluar un op matematica en un
 string
                self.operacion=""#volver a inicializar
                self.limpiarPantalla()
                self.mostrarEnPantalla(resultado)
            #Si se presionó el botón de borrado, limpiar la pantalla
            elif texto==u"\u232B":
                self.operacion=""
                self.limpiarPantalla()
        #Mostrar texto
        else:
            self.operacion+=str(texto)
            self.mostrarEnPantalla(texto)
        return
```

Fig. 7. Cuarta parte del codigo de la CArculadora en Python

```
    #Borra el contenido de la pantalla de la calculadora
    def limpiarPantalla(self):
        self.pantalla.configure(state="normal")

        self.pantalla.delete("1.0", END)#borra (1.0)fila y columna repectivamente[borra
mo s desde el inicio hasta]-> END= indice de text
        self.pantalla.configure(state="disabled")#Vuelve a disabled
        return

```

Fig. 8. Quinta parte del codigo de la CArculadora en Python

```
    #Muestra en la pantalla de la calculadora el contenido de las operaciones y los res
ultados
    def mostrarEnPantalla(self, valor):
        self.pantalla.configure(state="normal")
        self.pantalla.insert(END, valor)#inster=insertar texto en la pantalla
        self.pantalla.configure(state="disabled")
        return


ventana_principal=Tk()
calculadora=Interfaz(ventana_principal)
ventana_principal.mainloop()
```

Fig. 9. Sexta parte del codigo de la CArculadora en Python

VI. CONCLUCION

- Se pudo modificar un algoritmo en lenguaje de programación Python el cual nos permite realizar las 4 operaciones básicas.
- Tkinter es excelente para aplicaciones GUI pequeñas y rápidas, y dado que se ejecuta en más plataformas que cualquier otro kit de herramientas GUI de Python, es una buena opción donde la portabilidad es la principal preocupación.

.

REFERENCES

[1] (***Wxpython - Empezando Con Wxpython | Wxpython Tutorial***, **n.d.**)***¿Qué es GUI (interfaz gráfica de usuario) ? - Definición en WhatIs.com***. **(n.d.). Retrieved July 18, 2020, from https://searchdatacenter.techtarget.com/es/definicion/GUI-interfaz-grafica-de-usuario**

[2] **Chazallet, S. (2015).** ***Python 3 Los fundametos del lenguaje***. **https://www.ediciones-eni.com/open/mediabook.aspx?idR=9103588b3f4c184b8f2497bcf75605a9**

[3] **Fatima, H. (2017).** ***The 6 Best Python GUI Frameworks for Developers***. **https://blog.resellerclub.com/the-6-best-python-gui-frameworks-for-developer/**

[4] **Mokhtar Ebrahim. (2018, January 22).** ***Ejemplos de la GUI de Python (Tutorial de Tkinter) - Like Geeks***. **https://likegeeks.com/es/ejemplos-de-la-gui-de-python/**

[5] ***Operadores Python ➞ Explicamos todos los operadores + Ejemplos reales***. **(n.d.). Retrieved July 18, 2020, from https://pythones.net/operadores-basicos-en-python/**

[6] ***PyQt - Wikipedia, la enciclopedia libre***. **(n.d.). Retrieved July 18, 2020, from https://es.wikipedia.org/wiki/PyQt**

[7] ***Python***. **(n.d.). python.org. Retrieved July 18, 2020, from https://es.wikipedia.org/wiki/Python#:~:text=Python es un lenguaje de,en menor medida%2C programación funcional.**

[8] ***wxpython - Empezando con wxpython | wxpython Tutorial***. **(n.d.). Retrieved July 19, 2020, from https://riptutorial.com/es/wxpython**