implementación de la lista Doble enlazada

class DoublyNode:

def \_init\_(self, value=None):

self.value = value

self.prev = None

self.next = None

class DoublyLinkedList:

def \_init\_(self):

self.head = None

self.tail = None

self.current\_node = None

def append(self, value):

new\_node = DoublyNode(value)

if not self.head:

self.head = new\_node

self.tail = new\_node

self.current\_node = new\_node

else:

self.tail.next = new\_node

new\_node.prev = self.tail

self.tail = new\_node

self.current\_node = new\_node

def delete(self, value):

current = self.head

while current:

if current.value == value:

if current.prev:

current.prev.next = current.next

if current.next:

current.next.prev = current.prev

if current == self.head:

self.head = current.next

if current == self.tail:

self.tail = current.prev

if current == self.current\_node:

self.current\_node = current.prev if current.prev else current.next

return True

current = current.next

return False

def move\_forward(self):

if self.current\_node and self.current\_node.next:

self.current\_node = self.current\_node.next

def move\_backward(self):

if self.current\_node and self.current\_node.prev:

self.current\_node = self.current\_node.prev

def current(self):

return self.current\_node.value if self.current\_node else None

Interfaz grafica con tkinter

import tkinter as tk

from ttkbootstrap import Style

class TextEditorApp:

def \_init\_(self, root):

self.root = root

self.root.title("Editor de Texto con Deshacer/Rehacer")

# Lista doblemente enlazada para el historial de texto

self.history = DoublyLinkedList()

self.text\_area = tk.Text(self.root, width=40, height=10)

self.text\_area.pack(pady=10)

# Botones

self.save\_button = tk.Button(self.root, text="Guardar estado", command=self.save\_state)

self.save\_button.pack(side=tk.LEFT, padx=5)

self.undo\_button = tk.Button(self.root, text="Deshacer", command=self.undo)

self.undo\_button.pack(side=tk.LEFT, padx=5)

self.redo\_button = tk.Button(self.root, text="Rehacer", command=self.redo)

self.redo\_button.pack(side=tk.LEFT, padx=5)

def save\_state(self):

text = self.text\_area.get("1.0", tk.END).strip()

if text: # Evita guardar un estado vacío

self.history.append(text)

def undo(self):

if self.history.current():

self.history.move\_backward()

text = self.history.current()

self.text\_area.delete("1.0", tk.END)

self.text\_area.insert(tk.END, text)

def redo(self):

if self.history.current():

self.history.move\_forward()

text = self.history.current()

self.text\_area.delete("1.0", tk.END)

self.text\_area.insert(tk.END, text)

if \_name\_ == "\_main\_":

root = tk.Tk()

style = Style(theme="flatly") # O cualquier tema que prefieras

app = TextEditorApp(root)

root.mainloop()