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()
```