UNIVERSIDAD NACIONAL
GUILLERMO BROWN

TRABAJO PRÁCTICO INTEGRADOR

Materia: Programación Avanzada.

Grupo: 17.

Tema: Conversor de Archivos Word ↔ PDF.

Profesor: Gianluca Piriz.

Integrantes: Matias Hajny, Nicolás Legunda, Nicole Quintana y Julieta Rofrano.

Introducción

En el marco del presente trabajo práctico integrador, se desarrolló una aplicación de escritorio utilizando Python, cuyo objetivo principal es automatizar la conversión de archivos entre los formatos Word (.docx) y PDF (.pdf). La solución fue diseñada aplicando los principios fundamentales de la Programación Orientada a Objetos (POO) y utilizando librerías específicas como docx2pdf y pdf2docx. Este proyecto busca no solo ofrecer una solución práctica para el manejo de documentos, sino también demostrar la aplicación efectiva de conceptos avanzados de programación.

¿Por qué elegimos este tema?

En el ambiente laboral en el que nos desempeñamos nos pareció una herramienta útil para el desarrollo y conversión de documentos.

Este programa brinda una solución rápida a la conversión de documentos tanto de formato docx a pdf como visceversa

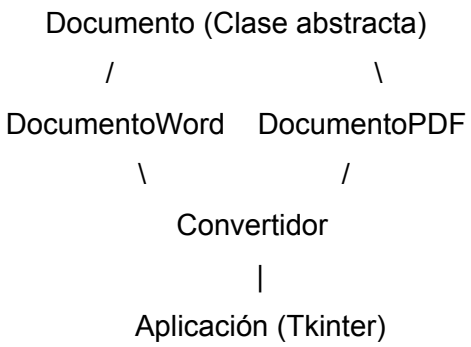
Descripción General

El sistema permite convertir archivos .docx a .pdf y viceversa. Se implementó una interfaz gráfica amigable con tkinter, que guía al usuario de forma intuitiva en el proceso de conversión. La arquitectura del programa se apoya en una jerarquía de clases, respetando los principios de diseño orientado a objetos para asegurar escalabilidad y mantenibilidad.

Objetivo

Desarrollar una herramienta sencilla y eficiente que permita a los usuarios automatizar la conversión de documentos entre formatos .docx y .pdf, minimizando errores humanos y reduciendo el tiempo destinado a esta tarea repetitiva.

Diagrama Conceptual



Componentes del Sistema

Componente	Descripción
Documento	Clase abstracta que define una interfaz común para todos los documentos.
DocumentoWord	Convierte archivos .docx a .pdf utilizando docx2pdf.
DocumentoPDF	Convierte archivos .pdf a .docx utilizando pdf2docx.
Convertidor	Ejecuta la conversión sin conocer el tipo de documento (polimorfismo).
Aplicación	Interfaz gráfica desarrollada con tkinter.

Clases Principales

Clase	Rol
Documento	Clase base abstracta con el método convertir().
DocumentoWord	Subclase que representa un documento Word y su conversión a PDF.
DocumentoPDF	Subclase que representa un documento PDF y su conversión a Word.
Convertidor	Controla la lógica de conversión (usa composición).
Aplicación	Administra la interfaz gráfica y la interacción con el usuario.

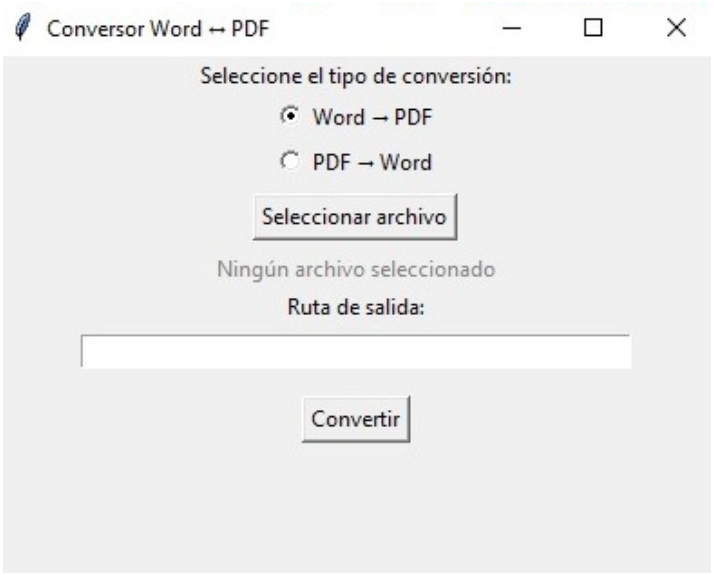
Princípios de POO Aplicados

Concepto	Implementación
Encapsulamiento	Cada clase gestiona internamente sus atributos y métodos.
Herencia	DocumentoWord y DocumentoPDF heredan de la clase abstracta Documento.
Polimorfismo	El método convertir() se comporta de forma distinta según la subclase.
Abstracción	La clase Documento no se puede instanciar directamente, solo se usa como base.

Interfaz Gráfica (GUI)

Características principales de la GUI implementada con tkinter:

- Permite seleccionar el tipo de conversión.
- Ofrece explorador de archivos para seleccionar documentos.
- Permite definir la ruta de salida.
- Muestra mensajes informativos ante errores o éxito.
- Fácil de usar y multiplataforma (aunque docx2pdf depende del SO).



Ventajas del Diseño

- Escalable: Puede adaptarse fácilmente para incluir nuevos formatos como .txt, .html, etc.
- Mantenible: Cada clase tiene una única responsabilidad clara.
- Portátil: Se puede utilizar tanto con GUI como desde la línea de comandos.

Consideraciones Técnicas

- La librería docx2pdf requiere que Microsoft Word esté instalado (solo funciona correctamente en Windows y macOS).
- pdf2docx puede fallar con archivos escaneados o con gran cantidad de imágenes.
- La interfaz está pensada para ser utilizada por usuarios sin conocimientos técnicos.

Conclusiones

El proyecto logra cumplir con el objetivo propuesto de automatizar la conversión de documentos de manera eficiente y con una experiencia de usuario intuitiva. Además, se consolidaron conocimientos clave de POO y uso de librerías externas en Python, logrando un producto funcional, profesional y extendible.

Posibles mejoras futuras

- Incorporar soporte para más tipos de archivos (HTML, TXT, ODT).
- Implementar arrastrar y soltar (drag & drop) para mejorar la UX.
- Añadir traducción automática de textos al momento de la conversión.
- Crear una versión web usando Flask o Django para mayor accesibilidad.