Reporte de Nuevas Funcionalidades y Mejoras para Utilidad PDF

Fecha: 22 de noviembre de 2025
Aplicación: Utilidad PDF con Streamlit
Tecnologías: Python, Streamlit, librerías open source

---

1. MEJORAS PARA FUNCIONALIDADES EXISTENTES

1.1 Extraer Páginas

Mejoras Recomendadas:

- **Vista previa de páginas:** Implementar thumbnails de las páginas usando `pdf2image` para que el usuario visualice qué páginas está extrayendo antes de confirmar
- **Selección múltiple inteligente:** Permitir rangos (ej: "1-5, 8, 10-12") además de selección individual con checkboxes
- **Validación de entrada:** Verificar que los números de página sean válidos y mostrar mensajes de error claros
- **Nomenclatura automática:** Generar nombres de archivo sugeridos automáticamente (ej: "documento_paginas_1-5.pdf")
- **Opción de descarga múltiple:** Si se extraen varias páginas como archivos separados, ofrecer descarga en ZIP

Implementación Streamlit:

```
# Ejemplo de selección por rangos
page_range = st.text_input("Páginas a extraer (ej: 1-5, 8, 10-12)")
# Mostrar preview con st.image() de las páginas seleccionadas
```

1.2 Reordenar Páginas

Mejoras Recomendadas:

- **Vista previa lado a lado:** Mostrar el orden actual vs. el orden nuevo antes de aplicar cambios
- **Drag & drop simulado:** Usar `streamlit-sortables` o una interfaz con botones ↑↓ para cada página
- **Plantillas de reordenamiento:** Ofrecer opciones predefinidas (invertir orden, páginas pares primero, páginas impares primero)
- **Validación de completitud:** Verificar que todas las páginas estén incluidas en el nuevo orden
- **Deshacer/Rehacer:** Mantener en `st.session_state` un historial de cambios

Implementación Streamlit:

```
# Usar session_state para mantener el orden
if 'page_order' not in st.session_state:
    st.session_state.page_order = list(range(1, num_pages + 1))
```

```
# Botones para mover páginas
col1, col2 = st.columns([3, 1])
```

1.3 Eliminar Páginas

Mejoras Recomendadas:

- **Previsualización antes de eliminar:** Mostrar qué páginas se eliminarán con thumbnails
- **Confirmación de acción:** Agregar un checkbox "Confirmo que quiero eliminar estas páginas" para evitar errores
- **Selección inversa:** Opción para "mantener solo estas páginas" en lugar de "eliminar estas páginas"
- **Eliminación por contenido:** Detectar y ofrecer eliminar páginas en blanco automáticamente
- **Estado de sesión:** Permitir marcar páginas para eliminar y aplicar cambios al final

Implementación Streamlit:

```
# Doble confirmación
pages_to_delete = st.multiselect("Selecciona páginas a eliminar", options)
if pages_to_delete:
    confirm = st.checkbox(" Confirmo que quiero eliminar estas páginas")
    if confirm:
        delete_button = st.button("Eliminar páginas")
```

---

2. NUEVAS FUNCIONALIDADES SUGERIDAS

2.1 Funcionalidades Básicas (Alta Prioridad)

A. Combinar/Unir PDFs

- **Descripción:** Subir múltiples PDFs y combinarlos en uno solo
- **Librería:** PyPDF2, pikepdf
- **Valor:** Funcionalidad fundamental en cualquier utilidad PDF
- **Dificultad:** Baja

```
# Ejemplo con PyPDF2
from PyPDF2 import PdfMerger
merger = PdfMerger()
for pdf in pdf_files:
    merger.append(pdf)
merger.write("combined.pdf")
```

B. Dividir PDF (Split)

- **Descripción:** Dividir un PDF en múltiples partes por número de páginas o rangos
- **Opciones:**

  - Dividir cada N páginas
  - Dividir en X partes iguales
  - Dividir por rangos personalizados
- **Librería:** PyPDF2
- **Dificultad:** Baja

C. Rotar Páginas

- **Descripción:** Rotar páginas específicas 90°, 180° o 270°
- **Uso:** Corregir orientación de páginas escaneadas
- **Librería:** PyPDF2 (método `.rotate()`)
- **Dificultad:** Baja

D. Información del PDF

- **Descripción:** Mostrar metadatos (autor, título, creación, modificación, número de páginas, tamaño, versión PDF)
- **Librería:** PyPDF2, pikepdf
- **Valor:** Útil para auditoría y organización
- **Dificultad:** Muy baja

```
# Widgets Streamlit sugeridos
st.metric("Número de páginas", num_pages)
st.json(metadata_dict)
```

2.2 Funcionalidades Intermedias (Prioridad Media)

E. Añadir Marcas de Agua

- **Descripción:** Insertar texto o imagen como marca de agua
- **Opciones:** Posición, opacidad, rotación, fuente
- **Librería:** reportlab + PyPDF2
- **Dificultad:** Media

F. Añadir Numeración de Páginas

- **Descripción:** Insertar números de página con formato personalizable
- **Opciones:** Posición (superior/inferior, izquierda/centro/derecha), formato (1, 1/10, Página 1 de 10)
- **Librería:** reportlab + PyPDF2
- **Dificultad:** Media

G. Comprimir PDF

- **Descripción:** Reducir tamaño del archivo optimizando imágenes y calidad
- **Librería:** pikepdf, ghostscript (vía subprocess)
- **Valor:** Muy solicitado para compartir por email
- **Dificultad:** Media

```
# Usar pikepdf para compresión
import pikepdf
```

```
with pikepdf.open('input.pdf') as pdf:
    pdf.save('output.pdf', compress_streams=True)
```

H. Convertir Imágenes a PDF

- **Descripción:** Convertir JPG, PNG, etc. a PDF
- **Librería:** PIL/Pillow + reportlab
- **Dificultad:** Baja-Media

I. Extraer Imágenes de PDF

- **Descripción:** Extraer todas las imágenes contenidas en un PDF
- **Librería:** PyMuPDF (fitz), pdf2image
- **Dificultad:** Media

2.3 Funcionalidades Avanzadas (Prioridad Baja/Opcional)

J. OCR - Reconocimiento de Texto

- **Descripción:** Convertir PDFs escaneados en PDFs con texto seleccionable
- **Librería:** pytesseract + pdf2image
- **Advertencia:** Requiere instalación de Tesseract OCR
- **Dificultad:** Alta

K. Proteger con Contraseña

- **Descripción:** Añadir contraseña de usuario y/o propietario, establecer permisos (copia, impresión)
- **Librería:** PyPDF2, pikepdf
- **Dificultad:** Media

```
# Ejemplo con PyPDF2
from PyPDF2 import PdfWriter
writer.encrypt(user_pwd="password", owner_pwd="admin_password")
```

L. Desbloquear PDF Protegido

- **Descripción:** Eliminar protección con contraseña (si se conoce)
- **Librería:** PyPDF2, pikepdf
- **Dificultad:** Baja

M. Convertir PDF a Imágenes

- **Descripción:** Exportar cada página como imagen (PNG, JPG)
- **Librería:** pdf2image, PyMuPDF
- **Dificultad:** Baja-Media

N. Búsqueda de Texto

- **Descripción:** Buscar texto en el PDF y mostrar páginas donde aparece
- **Librería:** PyMuPDF (fitz)
- **Dificultad:** Media

O. Editar Metadatos

- **Descripción:** Modificar autor, título, palabras clave, etc.
- **Librería:** PyPDF2, pikepdf
- **Dificultad:** Baja

---

3. MEJORAS GENERALES DE EXPERIENCIA DE USUARIO

3.1 Interfaz y Navegación

Sistema de pestañas: Usar `st.tabs()` para organizar funcionalidades en categorías:

```
tab1, tab2, tab3 = st.tabs(["Editar Páginas", "Combinar/Dividir", "Herramientas Avanz
```

Barra lateral para configuración: Aprovechar `st.sidebar` para opciones globales (calidad de export, formato de nomenclatura)

Breadcrumbs/Progreso: Mostrar pasos claros cuando una acción requiere múltiples fases:

```
st.progress(0.5)  # 50% completado
st.info("Paso 2 de 4: Selecciona las páginas a extraer")
```

3.2 Manejo de Archivos

Soporte multi-archivo: Usar `accept_multiple_files=True` en `st.file_uploader()`

Validación de archivos: Verificar que sean PDFs válidos y mostrar errores descriptivos

Límite de tamaño: Informar al usuario sobre límites de tamaño y páginas

Sesión persistente: Usar `st.session_state` para mantener el PDF cargado entre interacciones

3.3 Rendimiento

Caché: Usar `@st.cache_data` para operaciones costosas (lectura de PDF, generación de thumbnails)

```
@st.cache_data
def load_pdf(file):
    return PyPDF2.PdfReader(file)
```

Procesamiento asíncrono: Para archivos grandes, mostrar spinner:

```
with st.spinner('Procesando PDF...'):
    result = process_pdf(file)
```

Límites razonables: Establecer límites de páginas/tamaño para evitar timeouts

3.4 Manejo de Errores

Mensajes claros: Usar `st.error()`, `st.warning()`, `st.success()` apropiadamente

Try-except robusto: Capturar errores comunes (PDF corrupto, sin permisos, sin páginas)

Validación previa: Verificar condiciones antes de procesar (ej: que haya páginas seleccionadas)

3.5 Ayuda y Documentación

Tooltips: Usar `help=` en widgets para explicar opciones

```
st.number_input("Número de página", help="Ingresa el número de página a extraer (1-10
```

Expanders para ayuda: Sección expandible con ejemplos de uso

```
with st.expander("  ¿Cómo usar esta función?"):
    st.write("Ejemplo: Para extraer páginas 1, 3 y 5-10, escribe: 1,3,5-10")
```

Guía inicial: Mensaje de bienvenida con video o GIF demostrativo

---

4. CONSIDERACIONES TÉCNICAS

4.1 Librerías Recomendadas

Librería	Uso Principal	Instalación
PyPDF2	Manipulación básica (merge, split, rotate)	`pip install PyPDF2`
pikepdf	Compresión, encriptación, metadatos	`pip install pikepdf`
pdf2image	Convertir PDF a imágenes (thumbnails)	`pip install pdf2image` + Poppler
reportlab	Crear PDFs desde cero, marcas de agua	`pip install reportlab`
PyMuPDF (fitz)	Extracción de texto/imágenes, búsqueda	`pip install PyMuPDF`
pytesseract	OCR (opcional, requiere Tesseract)	`pip install pytesseract`
Pillow	Manipulación de imágenes	`pip install Pillow`

4.2 Estructura del Proyecto Sugerida

```
pdf-utility/
   app.py                  # Aplicación principal Streamlit
   requirements.txt        # Dependencias
   utils/
      pdf_operations.py  # Funciones de manipulación PDF
      validators.py      # Validación de archivos
      helpers.py         # Funciones auxiliares
   pages/                  # Multi-page app (opcional)
      1_Edit_Pages.py
```

```
    2_Merge_Split.py
    3_Advanced.py
temp/                  # Archivos temporales (añadir a .gitignore)
```

4.3 Seguridad

Limpieza de archivos temporales: Eliminar PDFs procesados después de la descarga

Validación de entrada: Nunca ejecutar código basado en nombres de archivo del usuario

Límites de recursos: Establecer timeouts y límites de memoria

---

5. ROADMAP SUGERIDO DE IMPLEMENTACIÓN

Fase 1: Consolidación (1-2 semanas)

- Implementar mejoras en funcionalidades existentes (vistas previas, validación)
- Añadir información del PDF
- Implementar sistema de pestañas

Fase 2: Funcionalidades Básicas (2-3 semanas)

- Combinar PDFs
- Dividir PDFs
- Rotar páginas
- Convertir imágenes a PDF

Fase 3: Funcionalidades Intermedias (3-4 semanas)

- Comprimir PDF
- Añadir marcas de agua
- Numeración de páginas
- Protección con contraseña

Fase 4: Funcionalidades Avanzadas (opcional)

- OCR
- Extracción de imágenes
- Búsqueda de texto
- Conversión PDF a imágenes

---

6. CONCLUSIONES Y RECOMENDACIONES FINALES

Prioridades Inmediatas:

1. **Mejorar las 3 funcionalidades existentes** con vistas previas y validación
2. **Implementar "Combinar PDFs"** (funcionalidad fundamental)
3. **Añadir "Información del PDF"** (rápido de implementar, muy útil)
4. **Reorganizar UI con pestañas** para escalar fácilmente

Diferenciadores Clave:

- **Vista previa visual:** Las thumbnails de páginas mejorarán significativamente la UX
- **Validación robusta:** Mensajes de error claros generan confianza
- **Combinación de operaciones:** Permitir encadenar acciones (ej: combinar → eliminar páginas → rotar → descargar)

Advertencias:

- pdf2image requiere **Poppler** instalado en el sistema (complejidad de despliegue)
- OCR con tesseract requiere instalación adicional (considerar si es crítico)
- Streamlit tiene **límites de upload** (default 200MB), documentar claramente

Métricas de Éxito Sugeridas:

- Tiempo promedio de procesamiento por operación
- Tasa de error en uploads
- Funcionalidades más utilizadas (analytics con Google Analytics o similar)

Próximos Pasos Recomendados:

1. Priorizar 3-5 funcionalidades del listado según necesidades de usuarios
2. Crear prototipos de UI con Streamlit para las funcionalidades elegidas

3. Implementar sistema de feedback del usuario dentro de la app
4. Considerar despliegue en Streamlit Cloud para acceso web