Documentación Técnica – opoTest

Creado por	D Daniel Bustos
Fecha de creación	@23 de noviembre de 2025 17:34
Categoría	
Última edición por	D Daniel Bustos
Fecha de última actualización	@23 de noviembre de 2025 17:34

📘 Documentación Técnica – opoTest

Conversor HTML → PDF basado en Electron

Índice

1. Arquitectura General del Sistema

Basado en Electron, el sistema está dividido en dos procesos:

✔️ Main Process (Node.js)

Gestiona:

- Ventanas
- Conversión PDF (`printToPDF`)
- Handlers IPC
- Diálogos nativos
- Servicios críticos

📄 Referencia: README.md → Arquitectura

✔️ Renderer Process (Browser)

Gestiona:

- Interfaz gráfica
- Drag & Drop
- Modales
- Estado de archivos
- Envío de comandos al Main

📄 Referencia: README.md → Renderer Process

---

2. Estructura del Proyecto

```
opoTest/
├── src/
│   ├── main/
│   │   ├── index.js
│   │   ├── windows/
│   │   │   ├── createMainWindow.js
│   │   │   └── createPDFWindow.js
```

```
│   │   ├── ipc/
│   │   │   ├── dialogHandlers.js
│   │   │   └── conversionHandlers.js
│   │   └── services/
│   │       └── pdfConverter.js
│   │
│   ├── renderer/
│   │   ├── index.html
│   │   ├── styles.css
│   │   ├── app.js
│   │   └── modules/
│   │       ├── fileManager.js
│   │       ├── modalManager.js
│   │       ├── dropzoneManager.js
│   │       └── uiController.js
│   │
│   ├── preload/
│   │   └── index.js
│   │
│   ├── utils/
│   │   ├── parser.js
│   │   └── template.js
│   │
│   └── shared/
│       └── constants.js
```

📄 Referencia: README.md – Estructura del Proyecto

---

3. Procesos Internos

3.1 Main Process

Responsable de:

- Crear ventanas
- Ejecutar `printToPDF()`

- Ejecutar conversiones múltiples
- Acceder al sistema de archivos (fs)
- Gestionar diálogos nativos

Módulos principales:

- `index.js`
- `PDFConverter`
- `conversionHandlers`
- `dialogHandlers`

3.2 Renderer Process

Responsable de:

- UI y validaciones ligeras
- Gestión de drag & drop
- Solicitud de títulos
- Actualización de estado
- Llamadas al API expuesto por preload (seguro)

Módulos principales:

- `UIController`
- `DropzoneManager`
- `FileManager`
- `ModalManager`

---

4. Comunicación IPC (Renderer → Main)

Flujo general

```
Renderer → window.api (preload) → ipcRenderer.invoke() → Main → Respu
esta
```

📄 Referencia: CASOS_DE_USO.md – CU-17 Comunicación IPC

📄 Referencia: README.md – Comunicación IPC

Canales IPC relevantes

- `select-folder`
- `select-files`
- `convert-multiple`
- `show-error`
- `show-success`

📄 Referencia: CASOS_DE_USO.md – Notas técnicas → Canales IPC

---

5. Módulos y Responsabilidades Técnicas

Main Process

Módulo	Responsabilidad
`index.js`	Orquestación general y creación de ventanas
`createMainWindow.js`	Ventana principal
`createPDFWindow.js`	Ventana oculta para `printToPDF()`
`dialogHandlers.js`	`dialog.showOpenDialog` y `showSaveDialog`
`conversionHandlers.js`	Entrada a la conversión vía IPC
`pdfConverter.js`	Lógica de conversión a PDF (núcleo técnico)

📄 Referencia: README.md – Responsabilidades Main Process

Renderer Process

Módulo	Responsabilidad
`app.js`	Bootstrap de los módulos
`fileManager.js`	Estado interno de archivos y metadatos
`modalManager.js`	Gestión de ventanas modales

Módulo	Responsabilidad
dropzoneManager.js	Drag & drop + validaciones locales
uiController.js	Interacción global UI

📄 Referencia: README.md – Responsabilidades Renderer

Utils & Shared

Módulo	Responsabilidad
parser.js	Parseo avanzado del HTML del test
template.js	Generación del HTML final para el PDF
constants.js	Config PDF, límites, canales IPC

📄 Referencia: CASOS_DE_USO – CU-12 / CU-13

---

6. Flujo Interno de Conversión HTML → PDF

Referencia: CASOS_DE_USO CU-04 (flujo detallado)

Referencia: README.md – Flujo de Conversión

Flujo técnico:

1. conversionHandlers recibe mensaje IPC
2. pdfConverter.convertMultiple() inicia loop
3. Para cada archivo:
   3.1 Validar si existe PDF → error si existe
   3.2 Leer archivo HTML con fs.readFileSync
   3.3 parser.parseHtml() → datos estructurados
   3.4 template.generateHtml() / generateBlankHtml()
   3.5 Cargar HTML en PDFWindow
   3.6 Ejecutar pdfWindow.webContents.printToPDF(A4 config)
   3.7 Guardar PDF en carpeta destino
4. Retornar lista de éxitos/errores

---

7. Lógica de Parseo y Templates

7.1 Parser (parser.js)

Referencia: CU-12 Parsear HTML

El parser:

- Crea DOM con **jsdom**
- Extrae:
  - Tema del examen
  - Preguntas
  - Respuestas y su estado (correcta, incorrecta, seleccionada...)
  - Justificaciones
  - Estadísticas
- Calcula porcentajes
- Devuelve objeto estructurado para el template

7.2 Template Generator (template.js)

Referencia: CU-13 Generar HTML para PDF

Funciones:

- `generateHtml()` → con respuestas
- `generateBlankHtml()` → sin respuestas
- Estilos CSS inline
- Codificación final del HTML para data URI

---

8. Gestión de Errores

Tipos:

- Archivos HTML inválidos
- Títulos duplicados

- Archivos no .html
- PDF existente
- Errores internos de conversión
- Cancelación de selección de carpeta
- Errores en DOM parsing

📄 Referencia: CU-10, CU-11 Manejo de errores de conversión

La conversión **no se detiene** por errores individuales → se registran y continúa.

---

9. Límites del Sistema

📄 Referencia: CU-08 / Notas Técnicas

- Máximo **10 archivos por zona**
- Extensión obligatoria: `.html`
- No sobrescribir PDFs existentes
- Títulos únicoss por zona

---

10. Testing y Cobertura

📄 Referencia: TESTING.md

Cobertura:

- **>85% statements**
- **>90% lines**
- Módulos críticos: **100% cobertura**

Módulos testeados:

- `pdfConverter.js` (100%)
- `parser.js`
- `template.js`
- `constants.js` (100%)
- Módulos renderer vía mocks

Tecnologías:

- Jest 29.7.0
- jsdom mocks
- fs mocks
- Istanbul para coverage

---

11. Integración con SonarQube

📄 Referencia: SONARQUBE.md

Incluye:

- Instalación de SonarScanner
- Uso por OS (`npm run sonar`, `npm run sonar:win`)
- Autenticación por token
- Análisis de métricas: bugs, vulnerabilidades, code smells, duplicación, cobertura

Integración para CI/CD (GitHub Actions incluida).

---

12. Tecnologías Utilizadas

Componente	Tecnología
Frontend UI	HTML, CSS, JS
Escritorio	Electron 28
Parser DOM	jsdom
Testing	Jest + Istanbul
IPC	contextBridge + ipcRenderer / ipcMain
Lógica	ES6 Modules (renderer), CommonJS (main)
PDF	Electron `printToPDF(A4)`