Documento Tecnico: OCR + IA para PDFs en Agentes

Version: 1.0 Fecha: 2026-02-21 Estado: Implementado

1) Objetivo

- Habilitar una carga de documentos PDF realmente util para el agente en /start/agents.
- Cubrir dos escenarios: PDFs con texto embebido y PDFs escaneados (imagen) mediante OCR automatico.

2) Problemas Detectados

- El parser inicial presentaba errores de worker en Next.js (fake worker failed).
- PDFs escaneados devolvian contenido vacio porque no traian capa de texto.
- La alternativa con tesseract.js fallo en runtime con MODULE_NOT_FOUND del worker interno.
- Algunos archivos quedaban guardados solo como metadata (sin content), por lo que el chat no podia responc con el documento.

3) Solucion Implementada

- Se centralizo el parseo en la ruta: app/api/agents/files/parse/route.ts.
- Se uso extraccion nativa de texto para PDF con pdfjs-dist.
- Si el texto viene vacio, se aplica OCR automatico con OpenAI Vision (fallback).
- Se removio tesseract.js del proyecto para evitar fallos de empaquetado/runtime.

4) Flujo Tecnico Final

- Frontend (FileUploadPanel) envia archivo a /api/agents/files/parse.
- Backend detecta tipo por extension, MIME y firma binaria %PDF.
- Para PDF: primero extractPdfText; si no hay texto, extractPdfTextWithOcr.
- La API retorna data: { name, type, content }.
- El frontend guarda brain.files con contenido real y ese contenido se usa en ChatTest.

5) Cambios por Archivo

- app/api/agents/files/parse/route.ts: parser robusto + fallback OCR con IA.
- app/start/agents/components/FileUploadPanel.tsx: carga, mensajes de error claros, estado de 'Cargando fuente', validaciones y soporte para archivos sin contenido indexado.
- app/start/agents/components/ChatTestPanel.tsx: envia solo archivos indexados, aviso visual si faltan contenidos.
- app/api/agents/chat-test/route.ts: mejor uso de contexto documental (extractos relevantes) y consumo por tokens reales.
- app/start/agents/create/page.tsx: correccion para guardar content (no solo size/type).
- app/start/agents/[id]/page.tsx: boton 'Traer desde URL' para contexto de empresa y mejoras de UX.

6) Consumo y Metricas

- Chat de prueba paso de credito fijo a consumo real por tokens cuando usa GPT.
- Se registra log de uso para el dashboard de consumo.
- El objetivo es que la prueba del agente tambien refleje consumo real del cliente.

7) Limitaciones Actuales

- OCR depende de OPENAI_API_KEY en el entorno.
- Para controlar latencia/costo, el OCR procesa un numero limitado de paginas por documento largo.
- Archivos historicos sin content deben re-subirse para indexarse correctamente.

8) Checklist de QA

- Subir PDF con texto embebido y validar respuesta con pregunta especifica.
- Subir PDF escaneado y validar que OCR lo indexa (puede tardar mas en primera corrida).
- Verificar en UI si un archivo queda como 'Sin contenido indexado'.
- Verificar que el chat de prueba descuente consumo y aparezca en uso/metricas.
- Probar 'Traer desde URL' y luego Guardar contexto en el agente.