Centro Interinstitucional en Ciencia de Datos

# PROVINCIA DE BUENOS AIRES

## Incorporación de herramientas de Inteligencia Artificial Generativa para la mejora de los procesos de asesoría jurídica en la Provincia de Buenos Aires

INFORME TÉCNICO

SEPTIEMBRE 2025

ÍNDICE

# Introducción

Este informe técnico corresponde al tercer hito del proyecto de *Incorporación de herramientas de Inteligencia Artificial Generativa para la mejora de los procesos de asesoría jurídica en la Provincia de Buenos Aires*, desarrollado por el Centro Interinstitucional en Ciencia de Datos (UBATEC), con el objetivo de desarrollar herramientas inteligentes que permitan una optimización de los recursos, brindar un mejor servicio jurídico a la Administración Pública provincial y obtener una mayor transparencia para la ciudadanía.

Durante este período se consolidaron las bases técnicas establecidas en los informes anteriores, avanzando significativamente en todas las verticales fundamentales del desarrollo: optimización del hardware donde corre la aplicación de manera local, el desarrollo de la arquitectura de agentes, la construcción de las bases de datos de normativa legal, el desarrollo del backend de la aplicación y el refinamiento de la interfaz de usuario en el frontend de la misma. Los avances logrados demuestran un paso sustancial en el despliegue de la solución en el entorno de producción.

Este informe da cuenta del trabajo realizado y los detalles técnicos de la solución desarrollada e implementada en la infraestructura del organismo.

# 1. Funcionalidades del Asistente Legal

Un asistente legal moderno que ayuda a los abogados a redactar y revisar textos legales utilizando un sistema de agentes inteligentes para el procesamiento de documentos legales, implementado con LangGraph y Server-Sent Events (SSE) para respuestas en tiempo real.

Se describe a continuación la estructura de flujo de información del sistema, estableciendo claramente cómo los datos atraviesan los diferentes componentes de la solución. Esta definición incluye la estructura de datos del estado del asistente, permitiendo un seguimiento preciso del procesamiento de cada consulta.

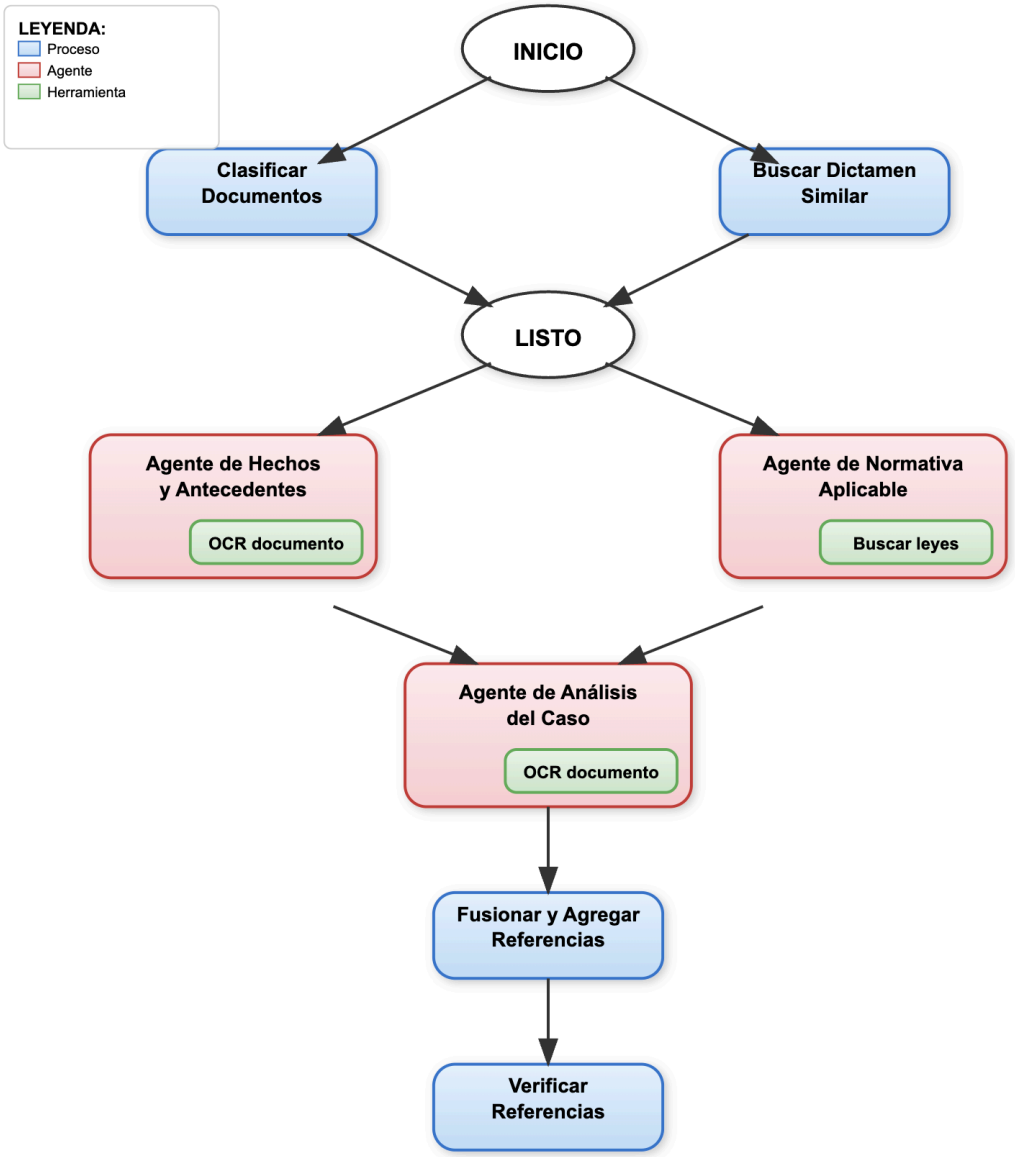
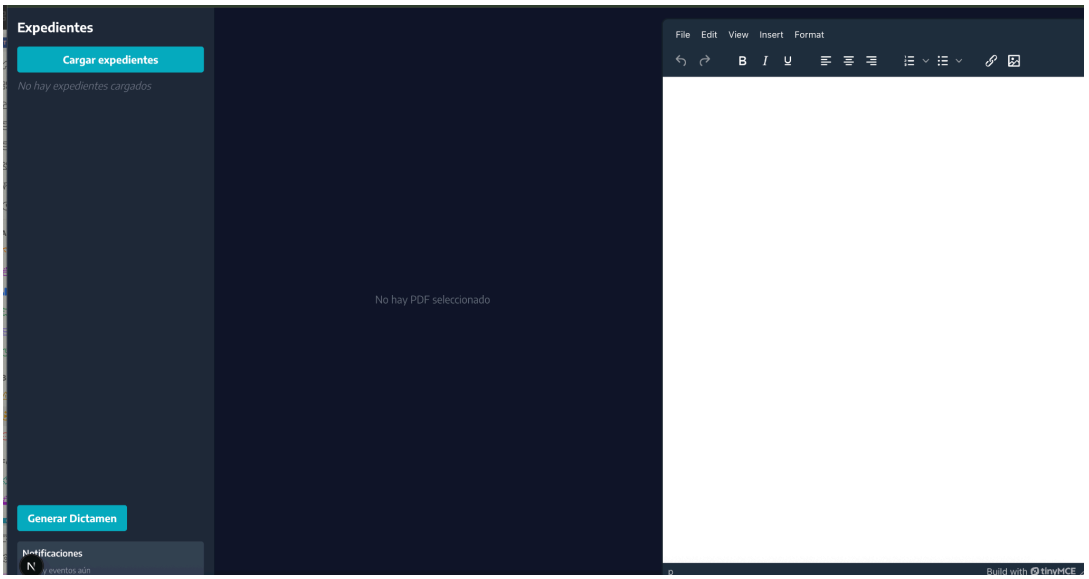
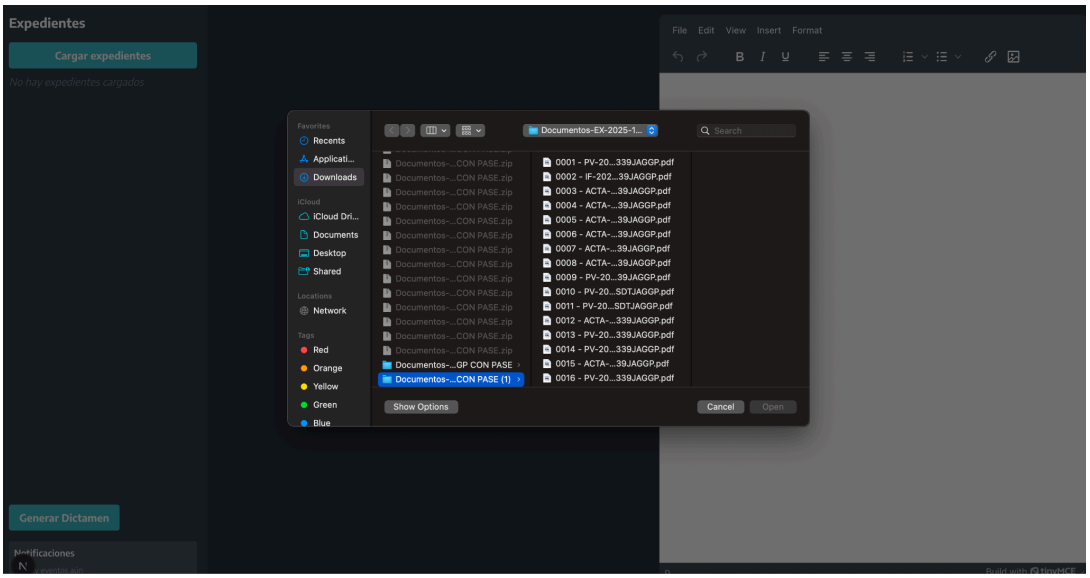


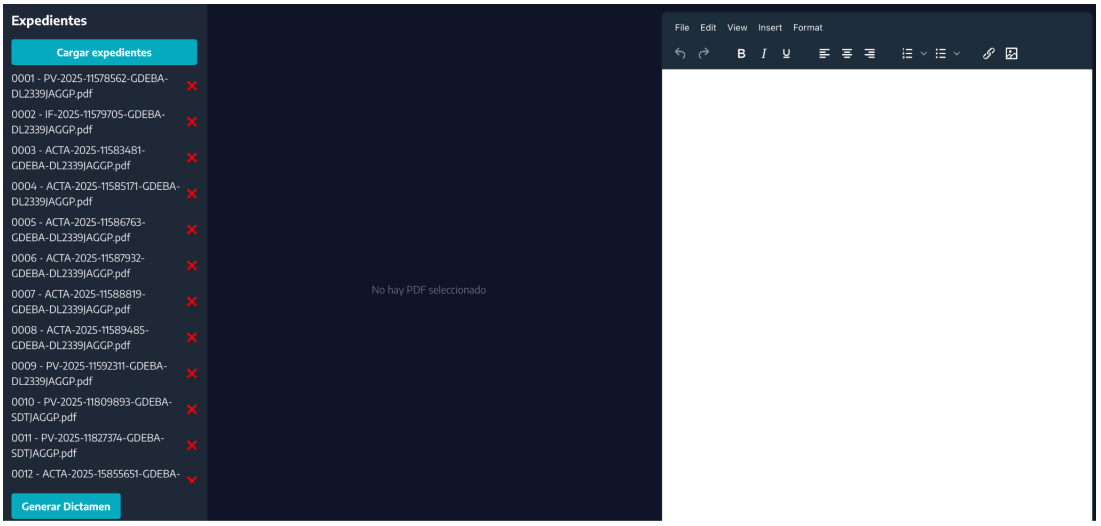

**Figura 1**. Esquema de funcionamiento del agente

## 1.1 Carga de expedientes

El sistema permite cargar expedientes en PDF en una interfaz de usuario desarrollada en React con TypeScript y Tailwind CSS.



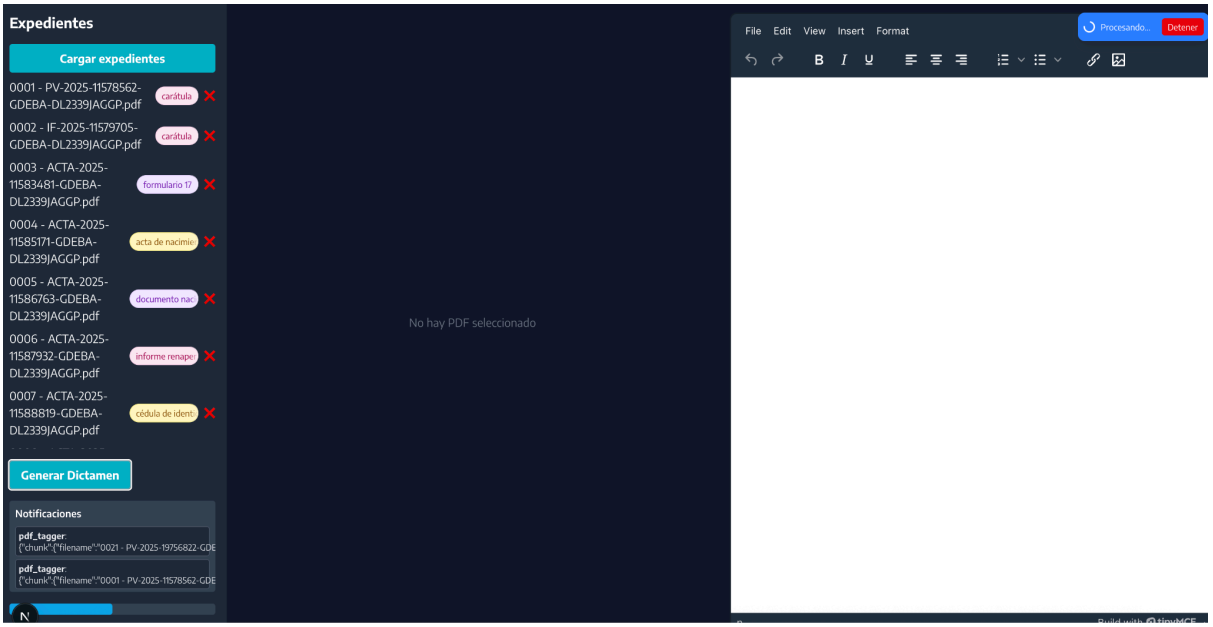



### 1.2 Etiquetado automático de documentos y OCR

El primer paso crítico del flujo de procesamiento es el etiquetado automático de documentos que componen los expedientes legales. Esta funcionalidad permite al sistema identificar automáticamente el tipo de documento y aplicar el tratamiento correspondiente de procesamiento y análisis. Los expedientes contienen una carátula, varios documentos sobre las personas involucradas, boletas de pago, pases administrativos, entre otros.

Por ejemplo, el sistema clasifica a los documentos en las siguientes categorías:

- carátula inicial del expediente
- acta de nacimiento
- constatación de parto
- acta de matrimonio
- acta de defunción
- documento nacional de identidad (DNI)
- cédula de identidad extranjera
- requerimiento renaper
- informe renaper
- declaración jurada de domicilio electrónico
- pase administrativo
- citación ciudadana
- requerimiento
- nota ológrafa
- boleta de pago
- pase a dictamen
- pase a subdirección técnica
- Entre otros tags




Se definió una estrategia integral para el procesamiento, extracción y transcripción de información contenida en formularios escaneados (mediante el uso de sistemas de OCR), un desafío técnico significativo dado el volumen de documentos digitalizados en el sistema jurídico provincial.

El sistema procesa los documentos adjuntados y realiza una extracción de datos estructurados a partir de texto e imágenes (por ejemplo, si es una cédula de identidad, extrae el nombre, apellido, fecha de nacimiento, dirección, etc. Si es un acta de nacimiento o defunción, extrae el nombre, apellido, fecha de nacimiento o defunción, lugar, etc.

### 1.3 Clasificación del tipo de caso prototípico

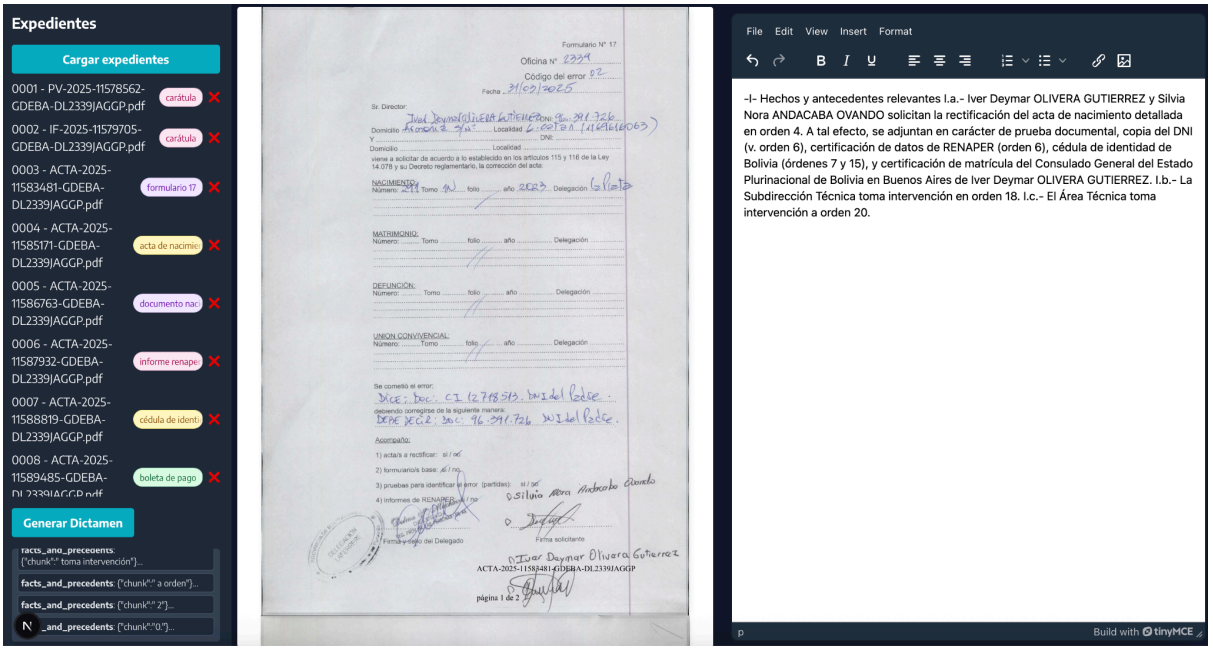
En segunda instancia, el sistema clasifica el tipo de caso legal que representa la carátula del expediente proporcionado. Los tipos de casos incluyen adición de apellido a un adolescente, adición de apellido a un menor, adición de apellido a un adulto, cambio de género a un menor, cambio de género a un adulto, rectificación de acta de defunción, rectificación de acta de defunción con cambio de licencia, rectificación de acta de nacimiento, reinscripción de acta de nacimiento, rectificación de varias actas (por ejemplo, de defunción y de nacimiento), rectificación de acta de matrimonio y rectificación de acta de unión convivencial.

En base al caso legal identificado por el sistema, el agente recupera el modelo estándar de dictamen prototípico que formará la base para la generación del dictamen jurídico. Este modelo está compuesto por las partes indicadas para la conformación de un dictamen: objeto, hechos y antecedentes relevantes , normativa aplicable, análisis del caso y conclusión.

## 1.4 Hechos y Antecedentes

Se implementó la funcionalidad para generar automáticamente la sección "Hechos y Antecedentes" de los dictámenes, una de las partes más críticas y que consume mayor tiempo en la elaboración manual. Esta implementación utiliza técnicas de OCR y “Structured outputs” para extraer y estructurar información relevante de los expedientes.

El agente especializado en hechos y precedentes se focaliza en la redacción de la primera sección de un dictamen, el cual es un resumen de los hechos y antecedentes relevantes del caso, construido a partir de los documentos en el expediente (los cuales poseen un número de orden, una etiqueta y un contenido). El agente sigue la estructura y formato de los ejemplos de dictámenes reales que le son adjuntados como ejemplos.




## 1.5 Normativa aplicable

En base a la sección previa del dictamen que contiene un resumen de los hechos y antecedentes relevantes del caso, el agente de normativa aplicable identifica la normativa pertinente. Esta segunda sección del dictamen es un listado citando los artículos aplicables al caso.

Como se ha indicado con mayor detalle en el 2° Informe de Avance, previamente el sistema ha procesado y construido una base de datos vectorial (Weaviate) para alojar normativa legal proveniente de múltiples fuentes:

- **Normativa nacional**: scraping de Infoleg
- **Normativa PBA:** scraping de normativa provincial específica
- **Manuales de procedimientos:** Documentación procedimental interna
- **Dictámenes anteriores:** Base de conocimiento histórica

Esta estructura permite una indexación eficiente y búsqueda semántica avanzada en todas las fuentes normativas relevantes.

Basándose en los hechos y antecedentes relevantes, el agente en primer lugar busca y recupera el texto de las normas relevantes en las colecciones previamente ingestadas de regulaciones, procedimientos y artículos de leyes mediante búsqueda semántica a partir de embeddings. Con estas normativas relacionadas al caso encontradas en los artículos de las leyes (título, número de artículo y texto) y en base al ejemplo prototípico proporcionado, el agente escribe la sección de normativa aplicable del dictamen.

### 1.6 Análisis del caso

En base a las secciones previas del dictamen acumulado que contienen los documentos del expediente, un resumen de los hechos y antecedentes y las normativas aplicables, el agente formula un análisis de caso siguiendo el formato del ejemplo proporcionado para el expediente actual y luego genera las conclusiones del dictamen.

# 2. Backend de la Aplicación

## 2.1 Estructura

Este directorio contiene el backend del Asistente Legal Copilot, implementado con FastAPI y LangGraph.

```
backend/
├── agent/                  # Agente LangGraph principal
│   ├── agent.py           # Configuración del grafo LangGraph
│   ├── main.py            # Servidor FastAPI con SSE
│   ├── utils/             # Utilidades del agente
│   │   ├── config.py      # Configuración del agente
│   │   ├── models.py      # Modelos de datos
│   │   ├── nodes.py       # Nodos del grafo LangGraph
│   │   └── state.py       # Estados del grafo
│   ├── requirements.txt   # Dependencias del agente
│   ├── README.md          # Documentación del agente
│   └── README_SSE.md      # Documentación SSE
├── main.py                # Servidor backend principal
├── pyproject.toml         # Configuración del proyecto
├── tests/                 # Tests del backend
│   ├── test_sse.py       # Tests SSE
│   └── test_noapi.py     # Tests sin API
└── README.md              # Este archivo
```

## 2.2 Componentes principales

### Agente LangGraph (agent/)

El componente principal del backend es el agente LangGraph que procesa documentos legales y genera respuestas inteligentes.

Características:

- Procesamiento de documentos PDF
- Extracción de texto e imágenes
- Etiquetado automático de documentos
- Streaming en tiempo real con SSE
- Soporte para múltiples servidores

Archivos clave:

- agent.py - Configuración del grafo LangGraph
- main.py - Servidor FastAPI con endpoints SSE
- utils/ - Utilidades y modelos del agente

**Servidor Backend Principal**

Servidor FastAPI principal que maneja las operaciones generales del sistema.

Endpoints:

- /health - Health check
- /api/convert-pdf - Conversión de PDFs

### 2.3 Stack tecnológico

- Python 3.12+ - Lenguaje principal
- FastAPI - Framework web
- LangGraph - Framework de agentes
- uv - Gestor de paquetes
- Pydantic - Validación de datos
- Uvicorn - Servidor ASGI
- Server-Sent Events - Streaming en tiempo real

# 3. Frontend de la Aplicación

### 3.1 Estructura

Interfaz principal del asistente legal con UI para:

- Chat con el agente
- Carga de documentos
- Visualización de respuestas en tiempo real

El Frontend del Asistente Legal Copilot está construido con Next.js 15, TypeScript y Tailwind CSS y se distribuye en el siguiente directorio:

```
frontend/
├── app/                    # Aplicación Next.js (App Router)
│   ├── api/                # API routes
│   │   ├── convert-pdf/    # Endpoint para conversión de PDFs
│   │   └── copilotkit/     # Endpoint para CopilotKit
│   ├── copilotkit/         # Página principal con CopilotKit
│   ├── test/               # Página de testing
│   ├── globals.css         # Estilos globales
│   ├── layout.tsx          # Layout principal
│   └── page.tsx            # Página principal
├── components/             # Componentes React
│   ├── agent-state.tsx     # Estado del agente
│   ├── interrupt.tsx       # Componente de interrupción
│   ├── sse-client.tsx      # Cliente SSE
│   ├── tool-call.tsx       # Llamadas a herramientas
│   └── ui/                 # Componentes UI
│       └── tooltip.tsx     # Tooltip personalizado
├── lib/                    # Utilidades
│   └── utils.ts            # Funciones utilitarias
├── public/                 # Archivos estáticos
├── package.json            # Dependencias
└── README.md               # Este archivo
```

Características:

- Interfaz moderna: Diseño limpio y profesional con Tailwind CSS
- Streaming en tiempo real: Integración con Server-Sent Events (SSE) para respuestas en tiempo real
- Procesamiento de PDFs: Carga y conversión de documentos PDF
- Componentes reutilizables: Biblioteca de componentes UI personalizados
- TypeScript: Tipado estático para mejor desarrollo
- Responsive: Diseño adaptativo para diferentes dispositivos

### 3.2 Componentes principales

#### SSE Client (components/sse-client.tsx)

Componente principal para manejar la comunicación SSE con el backend:

- Manejo de conexiones SSE
- Procesamiento de eventos en tiempo real
- Manejo de errores y reconexión
- Interfaz de usuario para mostrar respuestas

#### Agent State (components/agent-state.tsx)

Componente para mostrar el estado del agente:

- Estado actual del procesamiento
- Progreso de las operaciones
- Mensajes de estado

#### Tool Call (components/tool-call.tsx)

Componente para mostrar llamadas a herramientas:

- Visualización de herramientas utilizadas
- Resultados de las herramientas
- Estado de ejecución

### 3.3 Stack tecnológico

- Next.js 15 - Framework React con App Router
- React 19 - Biblioteca UI
- TypeScript - Tipado estático
- Tailwind CSS - Framework CSS
- Radix UI - Componentes de UI accesibles
- Lucide React - Iconos
- Server-Sent Events - Streaming en tiempo real

## Conclusiones

Los avances logrados durante este período demuestran el progreso sustancial en todas las verticales del proyecto, manteniendo el cronograma establecido y cumpliendo con los objetivos técnicos planteados.

**Logros destacados:**

- Desarrollo de una arquitectura de agentes inteligentes que cubren la gestión de todo el proceso de generación automática de dictámenes.
- Desarrollo de una aplicación escalable.
- Construcción de una base de datos de normativa legal robusta.
- Desarrollo de una interfaz de usuario integrada con estándares gubernamentales.
- Pruebas de integración con los sistemas existentes.
- Pruebas piloto con usuarios finales del organismo.

La implementación exitosa del sistema en servidores locales con optimizaciones de inferencia para mejorar la eficiencia de procesamiento, las pruebas de rendimiento y la implementación de componentes clave del sistema confirman la factibilidad de la solución para la generación de documentos legales de alta complejidad técnica como lo son los dictámenes jurídicos.

La solución desarrollada es un sistema integral basado en una arquitectura modular eficiente y que utiliza herramientas inteligentes de construcción y orquestación de agentes de IA. La combinación de técnicas avanzadas de búsqueda y recuperación semántica de información, el procesamiento multimodal de documentos legales, la generación de dictámenes asistida y una interfaz intuitiva de uso garantiza una solución funcional, adaptable y alineada con las necesidades del entorno jurídico.

Diego Fernández Slezak

Director del Proyecto