AUTOSELECT-X – SELECCIÓN AUTOMÁTICA DE EQUIPOS CON IA

Maestría en Computación para el Desarrollo de Aplicaciones Inteligentes
Escuela de Ingeniería de Sistemas y Computación
Trabajo Integrador III
Universidad del Valle
Cali, Colombia

Victor Anibal Murcia Calderon - 202402224
Fecha: 23 de Diciembre de 2025

Resumen

AutoSelect-X (AI EngiQuote) es una aplicación web orientada al sector Oil & Gas en Colombia que automatiza la extracción de requisitos técnicos a partir de documentos PDF de ingeniería (Hoja de Datos – HD, Especificación Técnica – ET y Manual de Requisitos – MR) para soportar la selección y cotización preliminar de equipos industriales, inicialmente bombas dosificadoras para sistemas de inyección de químicos. La solución integra un backend en Python (FastAPI), una base de datos relacional (PostgreSQL) y un pipeline de procesamiento documental que combina extracción estructural, normalización de unidades y asistencia por modelos de lenguaje (LLM) mediante un enfoque de Recuperación Aumentada por Generación (RAG), con capacidad de trazabilidad hacia el documento fuente.

Palabras clave

Extracción de información; PDFs de ingeniería; RAG; LLM; normalización de unidades; FastAPI; PostgreSQL; bombas dosificadoras; Oil & Gas.

Tabla de contenido

1. Información General del Proyecto

1.1 Identificación

Tabla 1.1. Identificación del proyecto.

Campo	Detalle
Nombre del proyecto	AutoSelect-X (AI EngiQuote)
Título formal	AutoSelect-X – Selección Automática de Equipos con IA (Enfoque inicial: bombas dosificadoras)
Institución	Universidad del Valle – Maestría en Ingeniería de Sistemas y Computación
Autor	Victor Murcia
Profesor/Curso	Luis Yovany Romo Portilla - Trabajo Integrador III
Profesor/Asesor	Victor Bucheli
Usuarios objetivo	Ingenieros de ventas/diseño, personal técnico en Oil & Gas (p. ej., especificaciones tipo Ecopetrol)
Repositorio	AutoSelectXV1 (GitHub) – versión desplegada en Render (commit base provisto por el autor)
Duración total del proyecto	Diez y ocho meses (18) meses

1.2 Planteamiento del alcance

El alcance del proyecto se centró en automatizar la lectura y extracción de requisitos técnicos desde PDFs de ingeniería asociados a paquetes de equipos (p. ej., skid de inyección de químicos), con énfasis en identificar de manera robusta el conjunto completo de bombas dosificadoras presentes (TAGS), así como parámetros críticos (caudal, presión de descarga, viscosidad, temperatura, servicio/fluido). La salida se estructura como JSON y se complementa con un proceso de normalización de unidades para habilitar la comparación y selección técnica posterior.

Fuera de alcance (en esta fase): (I) selección automática final por marca y referencia comercial con optimización multiobjetivo completa, (II) generación de propuesta comercial final con posible integración directa a ERP/CRM, (III) entrenamiento de modelos propios a gran escala (se prioriza consumo de APIs/modelos existentes y heurísticas de extracción).

1.3 Objetivo general

Desarrollar una aplicación web que procese documentos PDF de ingeniería y, mediante técnicas de extracción de información y asistencia por modelos de lenguaje con RAG, identifique requisitos técnicos de equipos (especialmente bombas dosificadoras) para apoyar la selección y generación de una cotización preliminar de forma trazable, reduciendo el tiempo y el error humano.

1.4 Objetivos específicos

1. Implementar autenticación, gestión de usuarios y control de acceso por roles para operación segura.
2. Implementar carga y pre-procesamiento de PDFs (lectura, limpieza, segmentación, extracción de texto/tablas) para aislar información relevante.
3. Implementar extracción asistida por LLM para identificar bombas y parámetros clave (caudal, presión, viscosidad, temperatura, servicio y TAG).
4. Normalizar unidades y valores, produciendo una salida consistente (p. ej., conversión a GPH/PSI).
5. Construir una interfaz conversacional por caso que permita consulta, visualización y auditoría de resultados (RAW vs normalizado).
6. Desplegar la solución en un entorno reproducible (local y nube), documentando instalación, configuración, operación y mantenimiento.

2. Contexto y Justificación

2.1 Descripción del problema o necesidad

En procesos de ingeniería y ventas técnicas en el sector Oil & Gas, la especificación de equipos se construye a partir de múltiples documentos PDF (HD/ET/MR) que pueden exceder decenas o cientos de páginas. La información relevante para la selección de equipos está distribuida en tablas, listas de TAGS, notas de ingeniería y anexos con formatos heterogéneos. Este escenario genera tres problemas recurrentes: (I) alta carga manual de lectura y búsqueda; (II) riesgo elevado de error por omisiones, interpretación incorrecta o conversión errónea de unidades; y (III) baja trazabilidad, dificultando justificar técnicamente una decisión de selección o una cotización frente a auditorías o revisiones internas.

2.2 Contexto de aplicación (sector, usuarios, entorno)

El proyecto se ubica en el contexto colombiano del sector petróleo y gas, con énfasis en sistemas de inyección de químicos (p. ej., inhibidores de corrosión, biocidas, demulsificantes). Los usuarios objetivo son ingenieros de ventas/diseño y personal técnico que requiere consolidar rápidamente parámetros técnicos de cada bomba del paquete (por TAG) para comparar alternativas comerciales y emitir una propuesta preliminar.

2.3 Justificación técnica y social del uso de IA

Técnicamente, la IA (en particular LLMs con enfoque RAG) permite convertir texto no estructurado y semiestructurado de documentos PDF en artefactos estructurados (JSON) con rapidez, reduciendo la fricción del análisis documental. Sin embargo, el sistema se concibe como asistente: la validación final de la información extraída recae en el usuario experto. Socialmente y académicamente, el proyecto aporta una herramienta para acelerar tareas repetitivas y mejorar la consistencia técnica, además de servir como caso de estudio aplicable a automatización documental en ingeniería.

2.4 Criterios de éxito

- Alta cobertura (recall) en detección de bombas y TAGS presentes en la especificación.
- Completitud de parámetros técnicos mínimos por bomba (flow, pressure, service/fluido, unidades).
- Normalización confiable: conversión a unidades estándar con reglas ancladas a unidades y contexto.
- Trazabilidad: capacidad de vincular valores extraídos con su origen documental (página/sección o fragmento recuperado).
- Operación estable en local y nube (Render), con configuración reproducible.

3. Diseño y Arquitectura de la Solución

3.1 Arquitectura general del sistema

AutoSelect-X se diseñó con una arquitectura modular en capas para desacoplar: (I) la experiencia web y endpoints de consulta, (II) la lógica de negocio y gestión de casos, (III) el pipeline de procesamiento documental (extracción y normalización), y (IV) la persistencia de datos. La implementación base utiliza FastAPI como framework ASGI en Python, con PostgreSQL como motor de persistencia transaccional para usuarios, roles y metadatos de casos.

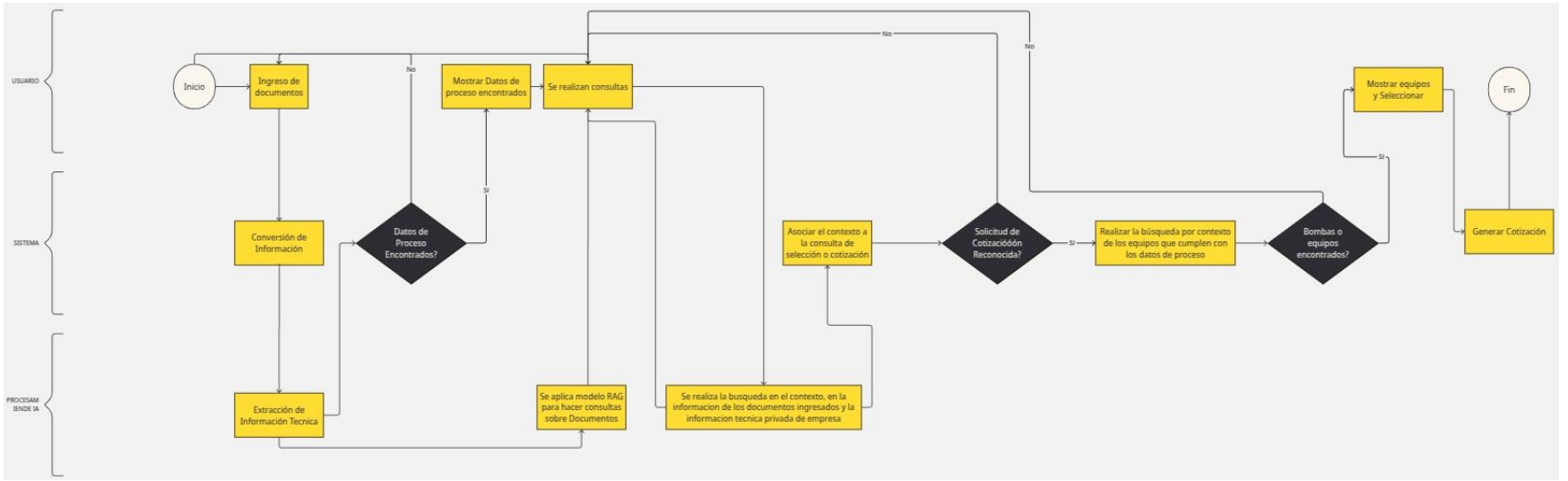
3.2 Componentes principales

- Frontend/Web UI: interfaz para autenticación, selección de casos, carga de PDFs y visualización de resultados.
- API Backend (FastAPI): routers para autenticación, gestión de casos, consulta conversacional y orquestación de pipeline.
- Pipeline de procesamiento de PDFs: lectura, limpieza, segmentación y extracción estructural (texto/tablas) para construir evidencias.
- Motor RAG/LLM: servicio que recibe una consulta y retorna una respuesta estructurada basada en contexto recuperado.
- Normalizador: módulo de estandarización y validación de valores técnicos (unidades, rangos, guardarraíles).
- Persistencia: PostgreSQL para entidades core (usuarios/casos) y almacenamiento de artefactos por caso en el filesystem del proyecto (outputs, cache, índices).

3.3 Flujo de proceso (SBPMN – descripción textual)

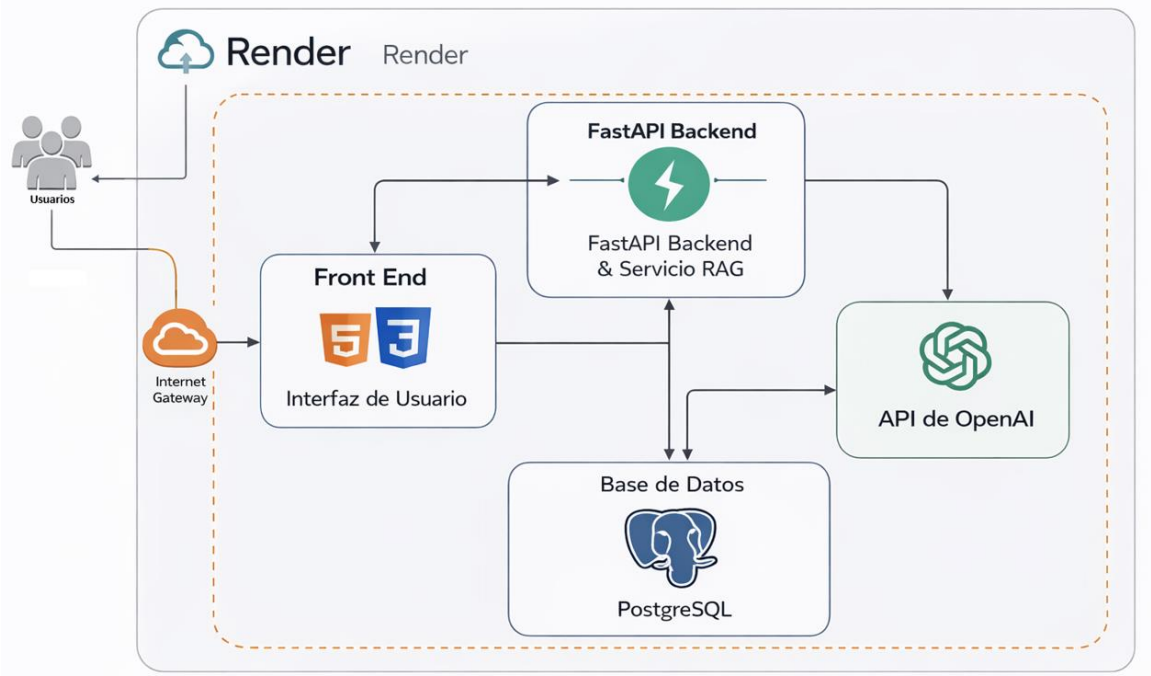
1. Inicio (Usuario): el usuario inicia el flujo de análisis para un caso.
2. Ingreso de documentos (Usuario): el usuario carga uno o más documentos (HD/ET/MR) asociados al caso.
3. Conversión de información (Sistema): el sistema almacena los PDFs en el repositorio del caso y ejecuta la conversión inicial (extracción de texto, limpieza básica y preparación del contenido para análisis).
4. Extracción de información técnica (Procesamiento IA): se ejecuta el pipeline de extracción para identificar datos de proceso y entidades técnicas relevantes (p. ej., bombas/TAGS, caudal, presión, viscosidad, temperatura, servicio).
5. Decisión: "¿Datos de proceso encontrados?" (Sistema):
   Sí: el sistema consolida y prepara los resultados para visualización.
   No: el flujo continúa hacia consultas asistidas, usando recuperación por contexto para intentar completar información faltante (sin bloquear al usuario).
6. Mostrar datos de proceso encontrados (Usuario/Sistema): el sistema presenta los parámetros extraídos (RAW y/o estructurados) y habilita su revisión.
7. Se realizan consultas (Usuario): el usuario formula consultas técnicas (por ejemplo: confirmar número de bombas, completar parámetros faltantes, validar TAGS o requisitos).
8. Aplicación del modelo RAG para consultas sobre documentos (Procesamiento IA): el sistema ejecuta el motor RAG para responder preguntas usando el contexto recuperado de los documentos cargados (y artefactos del caso).
9. Búsqueda en el contexto (Procesamiento IA): se realiza recuperación de fragmentos relevantes del caso (texto y/o tablas) y, cuando aplica, se complementa con información técnica privada disponible (catálogos/listas internas) para mejorar precisión.
10. Asociar el contexto a la consulta de selección o cotización (Sistema): el sistema empaqueta el contexto recuperado y lo vincula con la intención del usuario (consulta técnica vs. solicitud de selección/cotización).
11. Decisión: "¿Solicitud de cotización reconocida?" (Sistema):
    Sí: se activa el flujo de selección/cotización.
    No: el sistema retorna al ciclo de consultas (pasos 7–10) para seguir refinando extracción/validación.
12. Búsqueda por contexto de equipos que cumplen los datos de proceso (Sistema/Procesamiento IA): el sistema contrasta requisitos extraídos/normalizados contra el conjunto de equipos/catálogos disponibles (cuando esté habilitado) para proponer alternativas compatibles.
13. Decisión: "¿Bombas o equipos encontrados?" (Sistema):
    Sí: se preparan candidatos.
    No: se retorna a consultas para ajustar contexto, parámetros, unidades o criterios (pasos 7–12).
14. Mostrar equipos y seleccionar (Usuario): el sistema presenta los candidatos y el usuario selecciona la alternativa (o confirma la recomendación).

15. Generar cotización (Sistema): el sistema genera la salida de cotización (estructura preliminar) con base en el equipo seleccionado y los requisitos del caso.
16. Fin (Usuario/Sistema): se finaliza el flujo; los resultados quedan disponibles para consulta posterior y/o exportación.


USUARIO
SISTEMA
PROCESAMIENDE IA
Inicio
Ingreso de documentos
Conversión de Información
Extracción de Información Tecnica
Datos de Proceso Encontrados?
No
Si
Mostrar Datos de proceso encontrados
Se realizan consultas
Se aplica modelo RAG para hacer consultas sobre Documentos
Se realiza la busqueda en el contexto, en la informacion de los documentos ingresados y la informacion tecnica privada de empresa
Asociar el contexto a la consulta de selección o cotización
Solicitud de Cotizaciõõn Reconocida?
No
Si
Realizar la búsqueda por contexto de los equipos que cumplen con los datos de proceso
Bombas o equipos encontrados?
No
Si
Mostrar equipos y Seleccionar
Generar Cotización
Fin

3.4 Diagrama de despliegue (descripción)

La solución se despliega bajo un modelo cliente-servidor. Un navegador web consume endpoints del backend FastAPI (servido en un Web Service). El backend se conecta a una instancia de PostgreSQL mediante una cadena DATABASE_URL. Para ejecución de extracción asistida, el backend invoca un proveedor de LLM vía API (clave almacenada en variables de entorno). El pipeline escribe artefactos de caso en almacenamiento persistente del servicio (filesystem), lo cual demanda control estricto de cache por caso para evitar contaminación cruzada.



3.5 Tecnologías y frameworks

Tabla 3.1. Tecnologías y frameworks utilizados.

Capa	Tecnología	Propósito
Backend/API	Python + FastAPI + Uvicorn	Servicios ASGI, endpoints, orquestación de pipeline y RAG.
Persistencia	PostgreSQL + SQLAlchemy/psycopg2	Usuarios, roles, casos y metadatos; acceso transaccional.
Procesamiento PDF	pdfplumber + regex + utilitarios	Extracción robusta de texto/tablas, limpieza y segmentación.
IA/RAG	LightRAG (enfoque grafo) + LLM vía API	Recuperación de contexto y generación estructurada (JSON).
Despliegue	Render	Publicación web y ejecución del backend en nube.
Gestión ágil	Jira + Git/GitHub	Backlog, seguimiento Kanban/sprints y control de versiones.

3.6 Diseño de la interfaz de usuario

La interfaz de usuario de AutoSelect-X fue diseñada con un enfoque funcional, técnico y orientado al usuario ingeniero, priorizando la claridad visual, la trazabilidad del proceso y la reducción de carga cognitiva durante el análisis de documentos complejos de ingeniería.

El diseño responde a un principio fundamental del proyecto: la interfaz no debe ocultar la complejidad técnica, sino organizarla y hacerla auditable. La navegación se estructura de forma secuencial y guiada por casos, permitiendo al usuario comprender en todo momento en qué etapa del flujo se encuentra, qué información ya ha sido procesada y qué acciones están disponibles. La interfaz evita elementos superfluos y se apoya en componentes estándar, priorizando estabilidad, legibilidad y consistencia visual.

3.6.1 Pantalla de inicio

La pantalla de inicio constituye el punto de entrada operativo a AutoSelect-X. Su función principal es ofrecer una visión general del estado del sistema y de los casos activos, permitiendo al usuario retomar rápidamente trabajos previos o iniciar un nuevo análisis.

Desde esta pantalla se presentan indicadores básicos como:

- Número de casos creados.
- Casos recientes.
- Accesos directos a las acciones principales (crear caso, consultar casos existentes).

El diseño es deliberadamente sobrio, con énfasis en jerarquía visual clara y acceso rápido, evitando saturación de información. Esta decisión responde a la naturaleza técnica del usuario objetivo, quien prioriza eficiencia y rapidez sobre elementos gráficos complejos.

3.6.2 Pantalla de lista de casos.

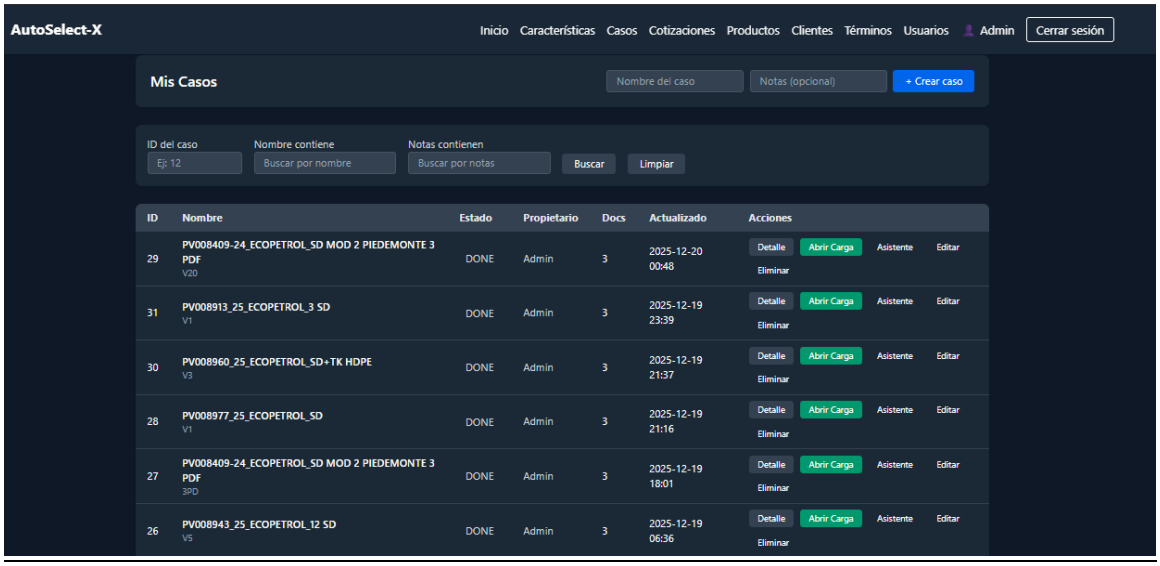
La pantalla de gestión de casos permite al usuario visualizar, crear, seleccionar y administrar los distintos casos de análisis. Cada caso representa una unidad lógica de trabajo independiente,

asociada a un conjunto específico de documentos técnicos (HD, ET, MR) y a sus respectivos artefactos de procesamiento.

Esta separación explícita por casos es clave para:

- Mantener aislamiento de información.
- Garantizar trazabilidad.
- Evitar contaminación cruzada entre análisis.

La interfaz presenta los casos en un listado estructurado, con información mínima pero suficiente (identificador, fecha, estado), y botones de acción claramente visibles. Esta organización facilita el trabajo iterativo y la revisión posterior de resultados.



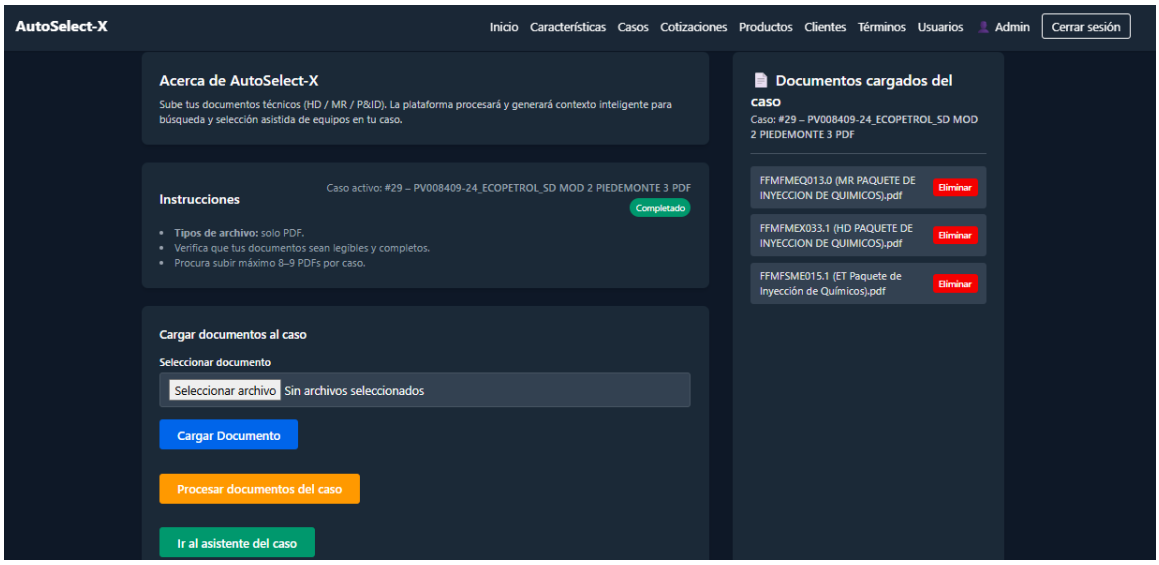
3.6.3 Pantalla de carga de documentos.

En esta pantalla el usuario adjunta los documentos PDF técnicos asociados a un caso específico. El diseño enfatiza la claridad del alcance de la acción, indicando explícitamente que los documentos cargados serán la única fuente de contexto técnico utilizada por el sistema para extracción y análisis.

La interfaz guía al usuario en el proceso de carga, mostrando:

- Estado de los archivos.
- Confirmación visual de carga exitosa.
- Asociación directa de los documentos con el caso activo.

Desde el punto de vista técnico, esta pantalla marca el inicio del pipeline de procesamiento documental, pero sin exponer su complejidad interna al usuario, manteniendo una experiencia controlada y predecible.



3.6.4 Pantalla de chat y pre-visualización de datos.

Esta pantalla constituye el núcleo interactivo del sistema. Aquí el usuario puede:

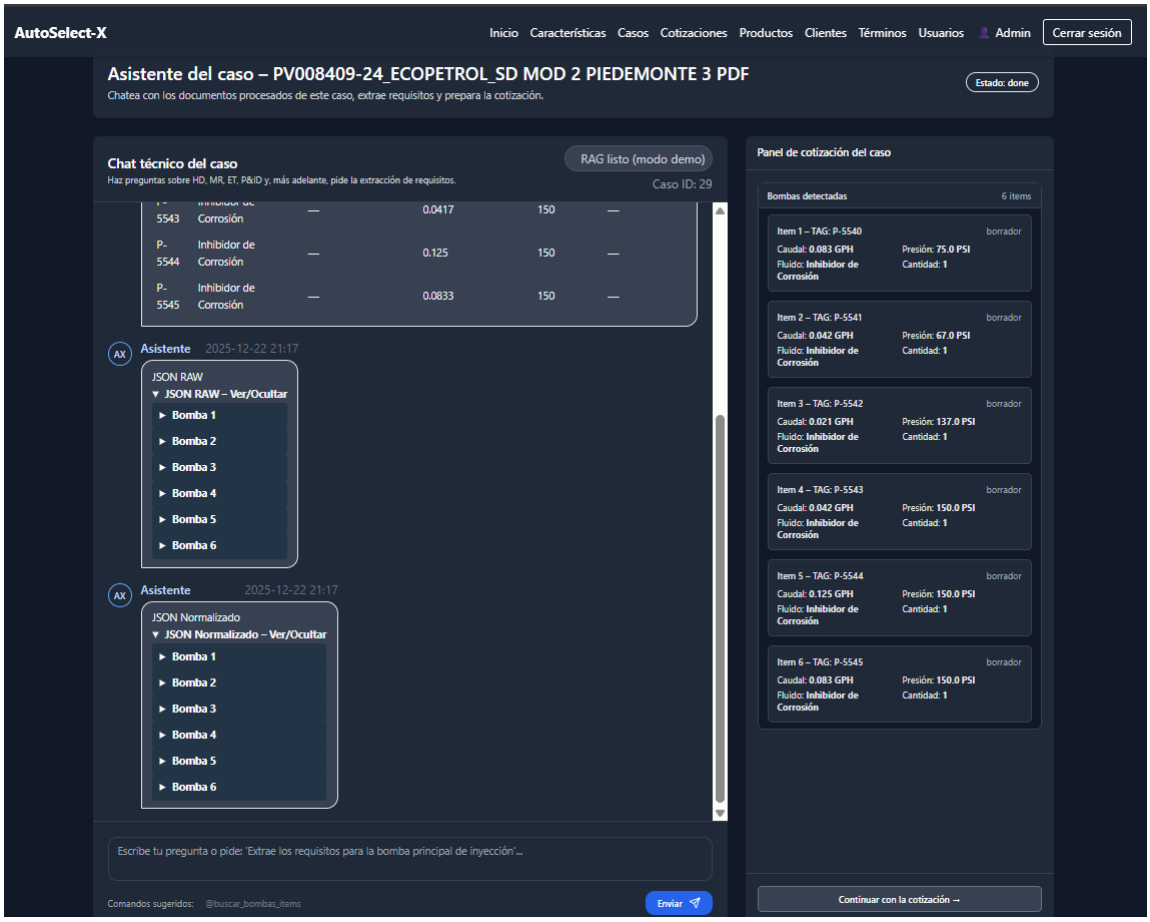
- Ejecutar la extracción automática inicial (lista completa de bombas).
- Formular consultas técnicas adicionales.
- Visualizar los resultados tanto en formato RAW como NORMALIZADO.

El diseño del chat no busca replicar una conversación informal, sino funcionar como una interfaz de consulta técnica estructurada, donde cada interacción genera resultados auditables y persistentes por caso.

La coexistencia de resultados RAW y NORMALIZADOS permite al usuario:

- Verificar la fidelidad del sistema al documento fuente.
- Identificar posibles ambigüedades o datos incompletos.
- Validar la correcta aplicación de reglas de normalización.

Este enfoque refuerza el carácter asistido del sistema y evita la percepción de “caja negra”.



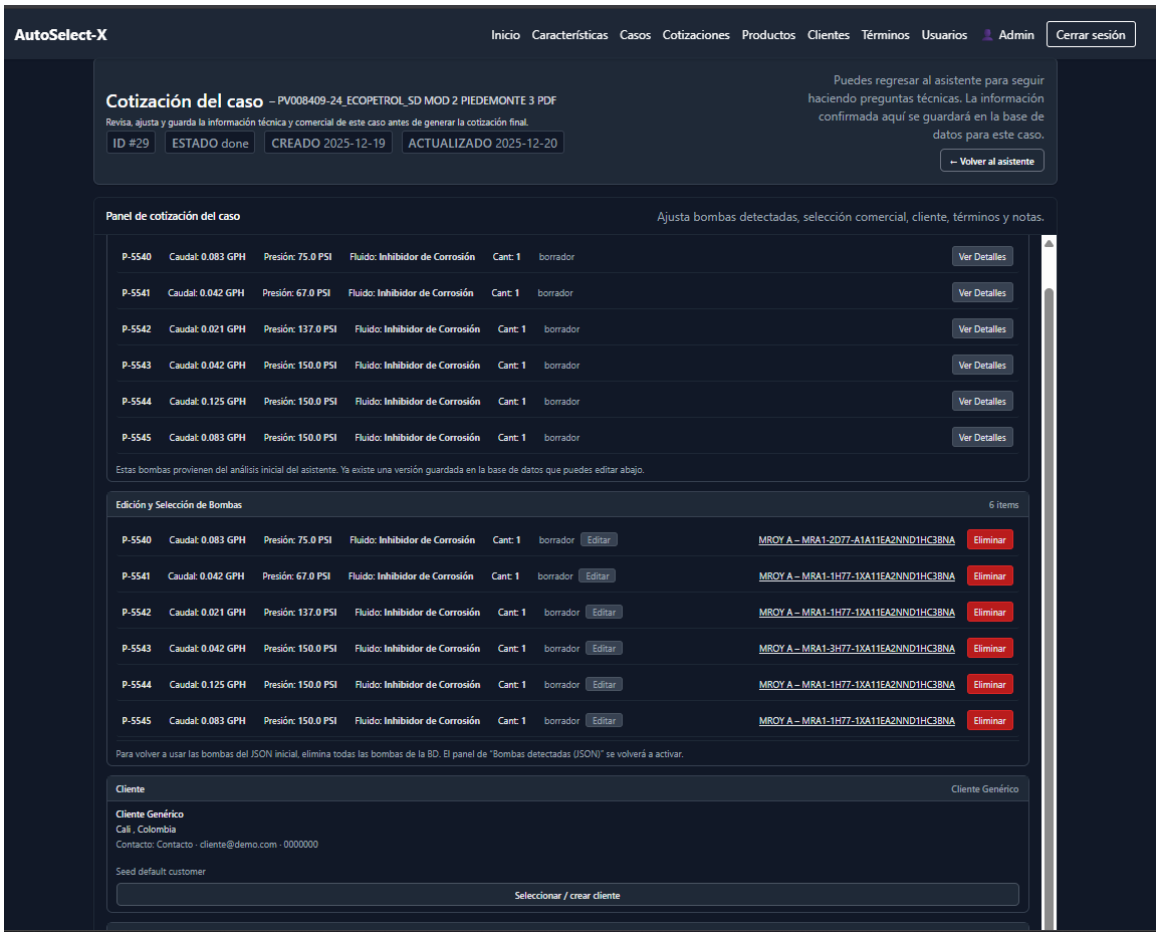
3.6.5 Pantalla del asistente de cotización.

Una vez consolidados los parámetros técnicos del caso, el sistema habilita el asistente de cotización. Esta pantalla organiza la información previamente extraída y normalizada, preparándola para su uso en procesos de selección y cotización preliminar.

El diseño se centra en presentar los datos técnicos de forma estructurada, destacando:

- Parámetros clave por bomba (caudal, presión, fluido, viscosidad).
- Consistencia de unidades.
- Campos incompletos o pendientes de validación.

La interfaz refuerza el rol del usuario como decisor técnico, ofreciendo soporte informativo sin imponer decisiones automáticas.



3.6.6 Pantalla de seleccionador del modelo de la bomba.

En esta pantalla se integran los resultados del análisis documental con la lógica de selección técnica. El usuario puede comparar alternativas de equipos que cumplen los requisitos del caso, apoyándose en los parámetros ya normalizados.

El diseño prioriza:

- Comparación clara entre alternativas.
- Visualización de compatibilidad técnica.
- Transparencia en los criterios utilizados para sugerir equipos.

Aunque el sistema puede proponer candidatos, la decisión final permanece explícitamente en manos del usuario, coherente con el enfoque ético del proyecto.

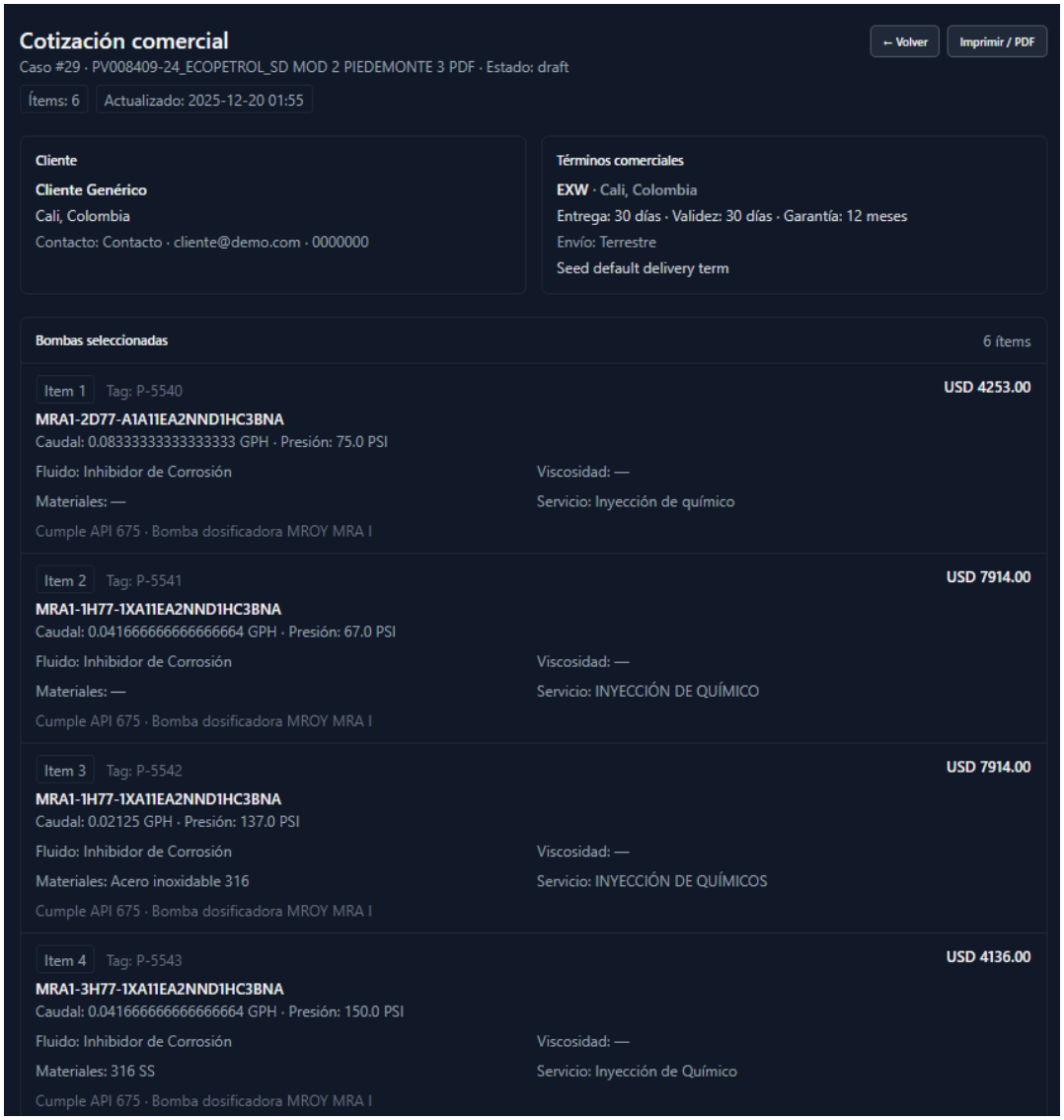
3.6.7 Pantalla de pre visualización de la cotización.

La etapa final del flujo corresponde a la pre-visualización de la cotización técnica, donde el usuario puede revisar la información consolidada antes de su exportación o uso externo.

Esta pantalla actúa como un punto de cierre del caso, mostrando:

- Datos técnicos finales.
- Equipos seleccionados.
- Coherencia general de la información presentada.

El diseño busca facilitar una revisión final rápida, minimizando errores antes de que la información sea utilizada en contextos comerciales o técnicos.



3.7 Estrategia de integración del modelo de IA

La integración de inteligencia artificial en la aplicación AutoSelect-X se realizó mediante una estrategia basada en el consumo de servicios externos de modelos de lenguaje de gran escala (LLM), integrados dentro de un pipeline RAG (Retrieval-Augmented Generation) unificado en el backend del sistema. Esta decisión respondió a la necesidad de contar con capacidades avanzadas de comprensión y generación de lenguaje técnico, sin incurrir en los costos computacionales y de mantenimiento asociados al entrenamiento de modelos propios.

En este contexto, AutoSelect-X utiliza la API de OpenAI como proveedor de modelos generativos, aprovechando su capacidad para interpretar texto técnico complejo proveniente de documentos de ingeniería (HD, ET y MR), y generar salidas estructuradas que apoyan la extracción de requisitos técnicos para la selección de equipos industriales, particularmente bombas dosificadoras. La elección de este enfoque permitió concentrar el esfuerzo del proyecto

en el diseño del pipeline de procesamiento documental, la normalización de datos técnicos y la trazabilidad de resultados, en lugar de la construcción de un modelo desde cero.

Desde el punto de vista arquitectónico, la integración de la IA se implementó de forma modular y centralizada en el backend, desarrollado en Python con FastAPI y desplegado en la plataforma Render. El backend actúa como intermediario entre la interfaz de usuario y el servicio de IA, encapsulando completamente la lógica de interacción con el modelo y evitando la exposición directa de claves o prompts al frontend. El motor RAG se encuentra unificado dentro del mismo servicio backend, lo que permite un control coherente del contexto, del almacenamiento de artefactos y del ciclo de vida de cada caso de análisis.

El flujo de integración del modelo de IA sigue las siguientes etapas lógicas:

1. Contextualización del caso: a partir de los documentos PDF cargados por el usuario, el sistema ejecuta un proceso de extracción y preprocesamiento que genera fragmentos textuales y metadatos organizados por caso.
2. Construcción del contexto RAG: el backend recupera los fragmentos relevantes asociados al caso, almacenados en el sistema de recuperación (LightRAG), garantizando que las consultas al modelo se realicen exclusivamente sobre información documental válida.
3. Generación del prompt estructurado: el contexto recuperado se combina con instrucciones técnicas y un esquema JSON predefinido, diseñado para forzar salidas estructuradas (lista de bombas, TAGS, parámetros de proceso).
4. Invocación del modelo de lenguaje: el prompt es enviado a la API de OpenAI mediante llamadas HTTP seguras, utilizando credenciales gestionadas a través de variables de entorno.
5. Validación y post-procesamiento: la respuesta generada es validada sintáctica y semánticamente, y posteriormente enviada al módulo de normalización, donde se aplican reglas de conversión de unidades y control de valores atípicos.
6. Interacción asistida con el usuario: los resultados son presentados en la interfaz como información asistencial, permitiendo al usuario revisar, validar y reutilizar los datos para procesos de selección o cotización preliminar.

Esta estrategia garantiza que el rol del modelo de inteligencia artificial en AutoSelect-X sea estrictamente asistencial y no decisorio, manteniendo siempre la validación técnica final en manos del usuario experto. Adicionalmente, el uso de un servicio externo de IA facilita la escalabilidad del sistema, permite la actualización transparente del modelo subyacente y asegura acceso a tecnología de última generación sin afectar la arquitectura general de la aplicación.

3.8 Gestión de cache, reproducibilidad y trazabilidad

Una de las consideraciones arquitectónicas más críticas en AutoSelect-X es la gestión del estado y de los artefactos intermedios generados por el pipeline de procesamiento documental y por el motor RAG. A diferencia de aplicaciones puramente transaccionales, el sistema genera y persiste múltiples archivos derivados del análisis de los documentos técnicos, tales como:

- Texto limpio consolidado por caso (post-procesamiento de PDFs).
- Fragmentos semánticos (chunks) utilizados para recuperación contextual.
- Índices y estructuras internas del motor RAG (embeddings, grafos y metadatos).
- Resultados intermedios y finales en formato JSON (RAW y NORMALIZED).

Estos artefactos se organizan por identificador de caso, lo que permite aislar la información técnica de cada conjunto documental. Sin embargo, durante las fases de desarrollo y pruebas se identificó un riesgo estructural relevante: la reutilización de un mismo identificador de caso sin una limpieza previa del estado puede inducir contaminación cruzada de resultados. En este escenario, consultas posteriores podrían recuperar información técnica perteneciente a un caso previamente procesado, afectando directamente la confiabilidad del sistema.

Para mitigar este riesgo, AutoSelect-X incorpora una estrategia explícita de control de caché y reinicio del pipeline (Reset), la cual permite:

- Eliminar de forma controlada los artefactos persistentes asociados a un caso específico.
- Limpiar completamente el almacenamiento del motor RAG cuando se requiere reproducir pruebas desde cero.
- Garantizar que cada ejecución del pipeline opere exclusivamente sobre los documentos cargados en el contexto actual.

Este mecanismo de reset es fundamental para asegurar la reproducibilidad experimental, ya que permite repetir ejecuciones bajo las mismas condiciones de entrada y verificar de manera objetiva el impacto de cambios en prompts, heurísticas de extracción o reglas de normalización.

Desde el punto de vista de trazabilidad, la arquitectura por caso facilita mantener una relación clara entre:

- Documento fuente → fragmento recuperado → campo extraído → valor normalizado. Aunque no todos los valores extraídos cuentan aún con una referencia explícita a página o sección, la persistencia de artefactos por caso constituye una base sólida para auditoría técnica y futuras mejoras en trazabilidad fina.

3.9 Seguridad, escalabilidad y mantenimiento

Seguridad

AutoSelect-X implementa un esquema de seguridad acorde con aplicaciones web modernas orientadas a entornos empresariales. El acceso al sistema se controla mediante autenticación y gestión de roles, restringiendo funcionalidades críticas a usuarios autorizados.

- Las credenciales sensibles, incluyendo:
- Cadena de conexión a la base de datos PostgreSQL.
- Claves de acceso a servicios externos de modelos de lenguaje (LLM).

Se gestionan exclusivamente mediante variables de entorno (.env) y no son versionadas en el repositorio de código fuente. Esta práctica reduce el riesgo de exposición accidental y se alinea con buenas prácticas de seguridad en despliegues en la nube.

Adicionalmente, el sistema contempla validaciones básicas sobre los archivos cargados (tipo y tamaño de PDF) y sobre las entradas de consulta, reduciendo superficies de ataque comunes. En un entorno productivo, se recomienda el uso de HTTPS y políticas adicionales de endurecimiento (hardening) del servicio.

Escalabilidad

La arquitectura actual de AutoSelect-X está diseñada para un número moderado de usuarios concurrentes, coherente con el contexto académico y con un escenario realista de uso en ingeniería de ventas, donde el procesamiento documental es intensivo pero no masivo.

La escalabilidad se aborda principalmente mediante:

- Separación clara de responsabilidades entre interfaz, backend, pipeline de procesamiento y motor RAG.
- Despliegue del backend como servicio único en Render, simplificando la operación inicial.
- Persistencia desacoplada mediante una base de datos SQL administrada.

Si bien el pipeline de extracción y normalización se ejecuta de forma síncrona en la versión actual, la arquitectura deja abierta la posibilidad de evolucionar hacia procesamiento asíncrono, por ejemplo:

- Ejecución del pipeline como jobs independientes.
- Uso de colas de tareas para manejar múltiples casos en paralelo.
- Externalización del almacenamiento vectorial en un servicio dedicado.

Estas extensiones no fueron implementadas en esta fase por razones de alcance académico, pero el diseño no las bloquea.

Mantenimiento
El mantenimiento del sistema se ve facilitado por un diseño modular, en el cual los componentes críticos (pipeline, normalizador, prompts RAG y lógica de negocio) están claramente separados. Esta decisión permite:

- Ajustar o mejorar reglas de normalización sin modificar la interfaz de usuario.
- Iterar sobre prompts y estrategias de extracción sin reestructurar la aplicación completa.
- Ejecutar pruebas de regresión por caso, comparando resultados antes y después de cambios técnicos.

Como práctica recomendada, se plantea la versionación explícita de prompts y reglas de normalización, acompañada de métricas por caso (recall, completitud y normalización), lo que permitiría evaluar objetivamente la evolución del sistema a lo largo del tiempo.

4. Modelo o Servicio de Inteligencia Artificial

4.1 Enfoque utilizado: RAG (Retrieval-Augmented Generation

El núcleo inteligente de AutoSelect-X se implementa mediante un enfoque de Retrieval-Augmented Generation (RAG), el cual combina técnicas de recuperación de información con modelos de lenguaje de gran tamaño (LLM). En este enfoque, el modelo no responde a partir de conocimiento general, sino que genera sus respuestas utilizando exclusivamente el contexto técnico recuperado desde los documentos cargados por el usuario.

Este diseño es especialmente adecuado para documentos de ingeniería, donde la precisión técnica y la fidelidad al texto fuente son prioritarias. Al restringir el contexto a fragmentos relevantes del caso, el sistema reduce significativamente el riesgo de alucinaciones y mejora la coherencia técnica de las respuestas.

4.2 Construcción de dataset/colección documental

A diferencia de enfoques tradicionales de aprendizaje automático, AutoSelect-X no entrena un modelo propio. En su lugar, el “dataset” del sistema corresponde a la colección dinámica de documentos PDF de ingeniería cargados por cada caso.

El proceso de preparación incluye:

- Lectura e ingesta de PDFs reales (no documentos pre-procesados).
- Limpieza estructural del texto (eliminación de encabezados repetidos, normalización de saltos de línea).
- Detección y serialización de tablas técnicas cuando es posible.
- Segmentación del contenido en fragmentos semánticos de tamaño controlado.

El resultado es una colección de fragmentos estructurados que constituyen la base para la recuperación contextual y permiten localizar secciones relevantes asociadas a TAGs, caudales, presiones y otros parámetros críticos.

4.3 Estrategia de prompts y schemas

Durante el desarrollo se definió una estrategia de extracción tipo 80/20, orientada a maximizar la cobertura y estabilidad del sistema. En esta estrategia:

1. El sistema prioriza la extracción de todas las bombas del paquete, independientemente de la pregunta formulada por el usuario.
2. Una vez identificada la lista completa de equipos, se realizan consultas adicionales para enriquecer o completar campos faltantes.

El modelo de lenguaje está obligado a responder utilizando un esquema JSON estricto, que define campos obligatorios y opcionales para cada bomba. Esta restricción:

- Facilita la validación automática.
- Permite detectar inconsistencias o valores ausentes.
- Sirve como insumo directo para el proceso de normalización.

4.4 Embeddings, recuperación y almacenamiento

El motor RAG de AutoSelect-X utiliza una estrategia de recuperación basada en representaciones semánticas de los fragmentos documentales. En la implementación actual se emplea un framework tipo LightRAG, que gestiona internamente:

4.5 Normalización

Una etapa crítica posterior a la extracción es la normalización de valores técnicos, cuyo objetivo es transformar información heterogénea en datos comparables y confiables.

El normalizador implementa reglas específicas, entre ellas:

- Aceptar valores de caudal únicamente cuando están explícitamente asociados a unidades de flujo (GPH, LPH, GPD, LPD).
- Ignorar números que no cumplan criterios de contexto, evitando interpretar códigos contractuales o identificadores como parámetros de proceso.
- Resolver rangos o pares de valores de presión mediante reglas definidas (por ejemplo, seleccionar el valor máximo reportado).

Estas reglas no buscan “inventar” información faltante, sino preservar la integridad técnica, dejando campos como nulos cuando el documento no proporciona el dato de forma explícita.

4.6 Consideraciones éticas y sesgos

AutoSelect-X se concibe explícitamente como un sistema de apoyo a la decisión, no como un reemplazo del juicio ingenieril. Dado el impacto potencial de decisiones técnicas incorrectas en contextos industriales, las respuestas generadas por el sistema requieren siempre validación humana.

Para mitigar riesgos asociados a errores o sesgos del modelo de lenguaje, se adoptaron las siguientes estrategias:

- Uso de RAG con contexto restringido a documentos del caso.
- Salida estructurada y validación posterior.
- Persistencia de artefactos por caso para auditoría.
- Transparencia en campos no extraídos (valores nulos en lugar de inferencias).

Este enfoque promueve un uso responsable de la inteligencia artificial y refuerza la confiabilidad del sistema en entornos de ingeniería.

5. Gestión Ágil del Proyecto (SCRUM)

El desarrollo de AutoSelect-X se gestionó mediante prácticas ágiles soportadas por Jira como herramienta central para planificación, seguimiento y control. El proyecto se ejecutó en un contexto académico con un único desarrollador (Victor Murcia), manteniendo un marco de trabajo alineado a SCRUM (backlog, sprints, revisión y retrospectiva) y utilizando prácticas Kanban para la gestión diaria de tareas técnicas, mejoras y correcciones.

5.1 Backlog del producto (actualizado con subtareas)

La Tabla 5.1 consolida el Product Backlog con las Historias de Usuario (HU), sus tareas, criterios de aceptación y Definition of Done (DoD). Esta organización se usa como referencia funcional del alcance. Posteriormente, se incluye la trazabilidad directa con Jira mediante las claves AXC-XX.

Tabla 5.1. Product Backlog (Historias de Usuario, tareas, criterios de aceptación y DoD).

ID	Historia de Usuario	Tareas (Jira)	Criterios de Aceptación	DoD
HU1	Como usuario quiero acceder con mis credenciales a la aplicación para utilizar sus funcionalidades.	• Crear modelo User en SQLAlchemy • Crear tabla users en PostgreSQL • Crear formulario de login (index.html) • Implementar ruta /login en user routes.py • Validar credenciales y estado activo • Generar cookie de sesión • Redirigir a /upload tras autenticación exitosa	• El usuario se puede autenticar si está activo. • Se crea y usa una cookie de sesión. • Usuario inactivo no accede.	• Código probado en navegador. • Sesión funcional y segura. • Acceso restringido por estado.
HU2	Como administrador quiero activar o desactivar usuarios para gestionar acceso a la aplicación.	• Crear vista usuarios.html • Implementar ruta /usuarios protegida por rol • Mostrar usuarios, estados y roles • Botones para activar/desactivar y cambiar rol • Guardar cambios en PostgreSQL • Validar que solo admin accede	• Admin ve y modifica estado/rol. • Cambios aplican correctamente. • Usuario actualizado refleja su nuevo estado.	• Vista funcional. • Cambios persistidos y visibles. • Protección por rol validada.
HU3	Como administrador quiero que el acceso esté	• Proteger rutas por sesión • /upload, /usuarios, etc. • Validar rol al acceder a rutas	• Solo usuarios autenticados acceden a rutas protegidas.	• Pruebas manuales exitosas. • Protección

	protegido mediante autenticación, control de sesiones y roles diferenciados.	• Ocultar/mostrar navbar según rol • Implementar logout (eliminar cookie)	• Admin accede a secciones exclusivas. • Logout cierra sesión y redirige.	activa en producción local.
HU4	Como usuario quiero subir documentos PDF técnicos para que el sistema los analice.	• Crear formulario de carga (upload.html) • Implementar ruta /upload en upload_routes.py • Guardar archivos en outputs/cases/... /user_id • Validar que sean PDFs válidos • Mostrar archivos subidos al usuario • Opción para eliminar archivo durante sesión	• El usuario puede subir y ver sus archivos. • Solo se aceptan PDFs. • Archivos se vinculan a la sesión.	• Carga funcional y estable. • Archivos listos para procesamiento posterior.
HU5	Como usuario quiero que los archivos PDF cargados sean leídos, limpiados y preparados para análisis, sin usar IA.	• Leer archivos con pdfplumber • Eliminar caracteres erróneos, encabezados y pies • Detectar tablas y párrafos relevantes • Extraer texto plano y almacenarlo limpio • Organizar por secciones técnicas si aplica	• Archivos leídos sin error. • Texto limpio y accesible para procesamiento.	• Al menos 2 PDFs convertidos con éxito. • Texto usable y revisado.
HU 15	Como desarrollador quiero validar las tecnologías de forma previa, validando respuesta de modelos, para seleccionar el modelo base a usar.	• Instalar y configurar Ollama en Ubuntu • Hacer Pull de modelos locales (Ollama, y Phi3) • Ejecutar prompts técnicos para evaluar respuesta • Generar reporte comparativo con modelo de pago	• Se prueban al menos 2 modelos: uno local y uno de pago. • Se genera un reporte comparativo.	• Pruebas ejecutadas y documentadas. • Modelo base definido para continuar con el desarrollo.
HU6	Como usuario quiero que el sistema procese el texto para extraer automáticamente requisitos técnicos como caudal y presión usando IA.	• Tokenizar texto extraído • Generar embeddings con modelo • Indexar en FAISS • Integrar LLM • Consultar el LLM para detectar campos técnicos clave	• Al menos 3 campos técnicos extraídos con IA. • Resultados interpretables y reutilizables.	• Extracción validada con IA sobre al menos 2 documentos. • Estructura funcional y lista para visualización.

		• Estructurar datos como diccionario		
HU7	Como usuario quiero visualizar los requisitos técnicos extraídos desde el PDF para validar que la información sea precisa.	• Mostrar datos extraídos en upload.html • Adaptar formato visual limpio • Input • Marcar campos vacíos o incompletos	• Usuario ve los datos del PDF. • Puede validar si el análisis fue correcto.	• Visualización funcional en navegador. • Compatible con múltiples archivos.
HU8	Como usuario quiero que de forma automática se consulte la base de datos técnica, obteniendo sugerencias de bombas compatibles con los requisitos extraídos.	• Generar embeddings con SentenceTransformers • Indexar en FAISS local • Buscar por similitud semántica • Mostrar resultados compatibles	• Se devuelve al menos una sugerencia técnica. • Resultados coherentes con requisitos.	• Función probada localmente con éxito. • Resultados visuales e interpretables.
HU9	Como usuario quiero que el sistema me asista con respuestas técnicas, basadas en mis documentos, para entender mejor los resultados.	• Integrar modelo LLM local (DeepSeek o LLaMA) • Conectar flujo con FAISS o PDF como contexto • Generar resumen o explicación técnica por prompt	• El LLM responde con base en documentos cargados. • El usuario entiende mejor los datos técnicos.	• Respuesta coherente. • Contexto bien inyectado desde embeddings.
HU 10	Como usuario quiero editar datos clave, antes de cerrar una cotización.	• Mostrar campos editables en vista resumen • Validar cambios en frontend • Actualizar datos antes de generar PDF	• Usuario puede modificar requisitos antes de cerrar. • Cambios se guardan y se usan en la cotización.	• Edición validada. • Datos actualizados correctamente.
HU 11	Como usuario quiero descargar una cotización técnica, basada en los requisitos de la bomba sugerida.	• Crear plantilla base de PDF de cotización • Incluir resumen técnico y selección de equipo • Generar PDF y ofrecer para descarga	• Cotización clara, coherente y descargable. • Incluye datos técnicos relevantes.	• PDF funcional y testeado. • Compatible con múltiples entradas.
HU 12	Como usuario quiero acceder a la aplicación desde la	• Configurar despliegue en Google Cloud (Cloud Run o App Engine)	• App disponible desde	• URL accesible y estable. • Flujo

	nube para usarla sin instalar nada localmente.	• Conectar base de datos Cloud SQL • Activar HTTPS y dominio seguro	navegador. • Funcionalidades básicas activas.	funcional validado en nube.
HU 13	Como administrador quiero validar el funcionamiento técnico de la aplicación antes de su entrega final.	• Ejecutar pruebas E2E • Validar rutas, login, carga, extracción, cotización • Documentar errores y corregir	• Todas las rutas y funciones operativas. • Comportamiento validado por administrador.	• Pruebas completadas. • Log de errores corregido.
HU 14	Como administrador quiero contar con documentación técnica del curso, al finalizar el desarrollo.	• Crear README técnico (instalación, arquitectura) • Manual de usuario paso a paso • Documentación de flujos y casos de uso	• Documentos completos, claros y entregables. • Cubren casos de uso reales.	• Documentos listos para entrega. • Evaluados por jurado o supervisor.

5.2 Trazabilidad del backlog en Jira

La Tabla 5.2 lista las HU registradas en Jira (proyecto AXC) y las ordena según la secuencia funcional del backlog. La Tabla 5.3 desglosa tareas y subtareas asociadas, manteniendo el mismo orden por HU.

Tabla 5.2. HU registradas en Jira (ordenadas según backlog).

HU No.	Key (Jira)	Título (Jira)	Sprint	Estado	Prioridad	Creada	Actualizada
1	AXC-2	HU1: Como usuario quiero acceder con mis credenciales a la aplicación para utilizar sus funcionalidades.	Sprint 1	Finalizada	Medium	29/may/25	29/may/25
2	AXC-3	HU2: Como administrador quiero activar o desactivar usuarios para gestionar el acceso a la aplicación.	Sprint 1	Finalizada	Medium	29/may/25	29/may/25

3	AXC-5	HU3: Como administrador quiero que el acceso esté protegido mediante autenticación, control de sesiones y roles diferenciados.	Sprint 2	Finalizada	Medium	29/may/25	29/may/25
4	AXC-6	HU4: Como usuario quiero subir documentos PDF técnicos para que el sistema los analice.	Sprint 2	Finalizada	Medium	29/may/25	29/may/25
5	AXC-7	HU5: Como usuario quiero que los archivos PDF cargados sean leídos, limpiados y preparados para análisis, sin usar IA.	Sprint 3	Finalizada	Medium	29/may/25	25/jun/25
15	AXC-75	HU15: Evaluación y definición del modelo LLM a usar.	Sprint 4, Sprint 5	Finalizada	Medium	29/may/25	03/oct/25
6	AXC-8	HU:6 Como usuario quiero que el sistema procese el texto para extraer automáticamente requisitos técnicos como caudal y presión usando IA.	Sprint 5	Finalizada	Medium	29/may/25	03/oct/25
7	AXC-9	HU:7 Como usuario quiero visualizar los requisitos técnicos	Sprint 5	Finalizada	Medium	29/may/25	24/oct/25

		extraídos desde el PDF para validar que la información sea precisa.					
8	AXC-10	HU:8 Como usuario quiero que de forma automática se consulte la base de datos técnica, obteniendo sugerencias de bombas compatibles con los requisitos extraídos.	Sprint 7	Finalizada	Medium	29/may/25	19/dic/25
9	AXC-11	HU:9 Como usuario quiero que el sistema me asista con respuestas técnicas, basadas en mis documentos, para entender mejor los resultados.	Sprint 6	Finalizada	Medium	29/may/25	01/dic/25
10	AXC-12	HU:10 Como usuario quiero editar datos clave, antes de cerrar una cotización.	Sprint 6	Finalizada	Medium	29/may/25	10/dic/25
11	AXC-13	HU:11 Como usuario quiero descargar una cotización técnica, basada en los requisitos de la bomba sugerida.	Sprint 8, Sprint 9	Finalizada	Medium	29/may/25	19/dic/25
12	AXC-14	HU:12 Como usuario quiero acceder a la aplicación desde	Sprint 8, Sprint 9	Finalizada	Medium	29/may/25	19/dic/25

		la nube para usarla sin instalar nada localmente.					
13	AXC-15	HU13: Como administrador quiero validar el funcionamiento técnico de toda la aplicación antes de su entrega final.	Sprint 9	Finalizada	Medium	29/may/25	19/dic/25
14	AXC-16	HU14: Como administrador quiero contar con documentación técnica de usuario, al finalizar el desarrollo.	Sprint 9	Finalizada	Medium	29/may/25	19/dic/25
17	AXC-84	HU17: Como usuario quiero validar y mejorar la extracción de información técnica desde los documentos, probando nuevos métodos de parseo y consulta RAG para obtener resultados más precisos	Sprint 10	Finalizada	Medium	24/oct/25	01/dic/25

Tabla 5.3. Tareas y subtareas por HU (trazabilidad Jira).

HU No.	**HU Key**	**Item Key**	**Título (tarea/subtarea)**	**Tipo**	**Sprint**	**Estado**
1	AXC-2	AXC-17	Crear tabla users en PostgreSQL	Subtarea	Tablero Sprint 1	Finalizada
1	AXC-2	AXC-18	Crear formulario de login (index.html)	Subtarea	Tablero Sprint 1	Finalizada
1	AXC-2	AXC-19	Implementar ruta /login	Subtarea	Tablero Sprint 1	Finalizada

1	AXC-2	AXC-20	Validar credenciales y estado activo	Subtarea	Tablero Sprint 1	Finalizada
1	AXC-2	AXC-21	Generar cookie de sesión	Subtarea	Tablero Sprint 1	Finalizada
1	AXC-2	AXC-22	Redirigir a /upload tras autenticación exitosa	Subtarea	Tablero Sprint 1	Finalizada
1	AXC-2	AXC-4	Crear modelo User	Subtarea	Tablero Sprint 1	Finalizada
2	AXC-3	AXC-23	Crear vista usuarios.html	Subtarea	Tablero Sprint 1	Finalizada
2	AXC-3	AXC-24	Implementar ruta /usuarios protegida por rol	Subtarea	Tablero Sprint 1	Finalizada
2	AXC-3	AXC-25	Mostrar usuarios, estados y roles	Subtarea	Tablero Sprint 1	Finalizada
2	AXC-3	AXC-26	Botones para activar/desactivar y cambiar rol	Subtarea	Tablero Sprint 1	Finalizada
2	AXC-3	AXC-27	Guardar cambios en PostgreSQL	Subtarea	Tablero Sprint 1	Finalizada
2	AXC-3	AXC-28	Validar que solo admin accede	Subtarea	Tablero Sprint 1	Finalizada
3	AXC-5	AXC-29	Proteger rutas por sesión (/upload, /usuarios, etc.)	Subtarea	Tablero Sprint 2	Finalizada
3	AXC-5	AXC-30	Validar rol al acceder a rutas	Subtarea	Tablero Sprint 2	Finalizada
3	AXC-5	AXC-31	Ocultar/mostrar navbar según rol	Subtarea	Tablero Sprint 2	Finalizada
3	AXC-5	AXC-32	Implementar logout (eliminar cookie)	Subtarea	Tablero Sprint 2	Finalizada
4	AXC-6	AXC-33	Crear formulario de carga (upload.html)	Subtarea	Tablero Sprint 2	Finalizada
4	AXC-6	AXC-34	Implementar ruta /upload en upload_routes.py	Subtarea	Tablero Sprint 2	Finalizada
4	AXC-6	AXC-35	Guardar archivos en outputs/session_files/{user_id}/	Subtarea	Tablero Sprint 2	Finalizada
4	AXC-6	AXC-36	Validar que sean PDF válidos	Subtarea	Tablero Sprint 2	Finalizada
4	AXC-6	AXC-37	Mostrar archivos subidos al usuario	Subtarea	Tablero Sprint 2	Finalizada
4	AXC-6	AXC-38	Opción para eliminar archivo durante sesión	Subtarea	Tablero Sprint 2	Finalizada
5	AXC-7	AXC-39	Leer archivos con pdfplumber	Subtarea	Tablero Sprint 3	Finalizada
5	AXC-7	AXC-40	Eliminar caracteres erróneos, encabezados y pies	Subtarea	Tablero Sprint 3	Finalizada
5	AXC-7	AXC-41	Detectar tablas y párrafos	Subtarea	Tablero	Finalizada

			relevantes		Sprint 3	
5	AXC-7	AXC-42	Extraer texto plano y almacenarlo limpio	Subtarea	Tablero Sprint 3	Finalizada
5	AXC-7	AXC-43	Organizar por secciones técnicas si aplica	Subtarea	Tablero Sprint 3	Finalizada
15	AXC-75	AXC-78	Ejecutar prompts técnicos para evaluar respuesta, tiempo y contexto	Subtarea	Tablero Sprint 4, Tablero Sprint 5	Finalizada
15	AXC-75	AXC-79	Generar reporte comparativo.	Subtarea	Tablero Sprint 4, Tablero Sprint 5	Finalizada
15	AXC-75	AXC-76	Instalar y configurar Ollama en Ubuntu.	Subtarea	Tablero Sprint 4, Tablero Sprint 5	Finalizada
6	AXC-8	AXC-44	Tokenizar texto extraído	Subtarea	Tablero Sprint 5	Finalizada
6	AXC-8	AXC-45	Generar embeddings con modelo seleccionado	Subtarea	Tablero Sprint 5	Finalizada
6	AXC-8	AXC-46	Indexar en DB Vectorial	Subtarea	Tablero Sprint 5	Finalizada
6	AXC-8	AXC-47	Integrar LLM	Subtarea	Tablero Sprint 5	Finalizada
6	AXC-8	AXC-48	Consultar el LLM para detectar campos técnicos clave	Subtarea	Tablero Sprint 5	Finalizada
6	AXC-8	AXC-49	Estructurar datos como diccionario	Subtarea	Tablero Sprint 5	Finalizada
7	AXC-9	AXC-50	Mostrar datos extraídos	Subtarea	Tablero Sprint 5	Finalizada
7	AXC-9	AXC-51	Adaptar formato visual limpio	Subtarea	Tablero Sprint 5	Finalizada
7	AXC-9	AXC-52	Marcar campos vacíos o incompletos	Subtarea	Tablero Sprint 5	Finalizada
8	AXC-10	AXC-53	Generar base de datos de bombas mRoyA	Subtarea	Tablero Sprint 7	Finalizada
8	AXC-10	AXC-54	Generar Script de confgurador de bombas mRoyA	Subtarea	Tablero Sprint 7	Finalizada
8	AXC-10	AXC-55	Generar Base de datos de bombas Seleccionadas o Configuradas	Subtarea	Tablero Sprint 7	Finalizada

8	AXC-10	AXC-56	Mostrar resultados compatibles	Subtarea	Tablero Sprint 7	Finalizada
9	AXC-11	AXC-57	Integrar modelo LLM con el RAG	Subtarea	Tablero Sprint 6	Finalizada
9	AXC-11	AXC-58	Conectar flujo del RAG por caso	Subtarea	Tablero Sprint 6	Finalizada
9	AXC-11	AXC-59	Extraer de forma automática las bombas solicitadas	Subtarea	Tablero Sprint 6	Finalizada
10	AXC-12	AXC-60	Mostrar campos editables en vista resume	Subtarea	Tablero Sprint 6	Finalizada
10	AXC-12	AXC-61	Validar cambios en frontend	Subtarea	Tablero Sprint 6	Finalizada
10	AXC-12	AXC-62	Actualizar datos de las bombas detectadas antes de generar PDF	Subtarea	Tablero Sprint 6	Finalizada
11	AXC-13	AXC-63	Crear plantilla base de PDF de cotización	Subtarea	Tablero Sprint 8, Tablero Sprint 9	Finalizada
11	AXC-13	AXC-64	Incluir resumen técnico y selección de equipo	Subtarea	Tablero Sprint 8, Tablero Sprint 9	Finalizada
11	AXC-13	AXC-65	Generar PDF y ofrecer para descarga	Subtarea	Tablero Sprint 8, Tablero Sprint 9	Finalizada
12	AXC-14	AXC-66	Configurar despliegue en Google Cloud (Cloud Run o App Engine)	Subtarea	Tablero Sprint 8, Tablero Sprint 9	Finalizada
12	AXC-14	AXC-67	Conectar base de datos Cloud SQL	Subtarea	Tablero Sprint 8, Tablero Sprint 9	Finalizada
12	AXC-14	AXC-68	Activar HTTPS y dominio seguro	Subtarea	Tablero Sprint 8, Tablero Sprint 9	Finalizada
13	AXC-15	AXC-69	Ejecutar pruebas E2E	Subtarea	Tablero Sprint 9	Finalizada

13	AXC-15	AXC-70	Validar rutas, login, carga, extracción, cotización	Subtarea	Tablero Sprint 9	Finalizada
13	AXC-15	AXC-71	Documentar errores y corregir	Subtarea	Tablero Sprint 9	Finalizada
14	AXC-16	AXC-72	Crear README técnico (instalación, arquitectura)	Subtarea	Tablero Sprint 9	Finalizada
14	AXC-16	AXC-73	Manual de usuario paso a paso	Subtarea	Tablero Sprint 9	Finalizada
14	AXC-16	AXC-74	Documentación de flujos y casos de uso	Subtarea	Tablero Sprint 9	Finalizada
17	AXC-84	AXC-85	Probar diferentes métodos de parseo y extracción de tablas en los documentos HD, MR y ET.	Subtarea	Tablero Sprint 10	Finalizada
17	AXC-84	AXC-86	Ejecutar pruebas con Docling y comparar resultados frente al pipeline previo.	Subtarea	Tablero Sprint 10	Finalizada

5.3 Definition of Done (DoD) - estándar aplicado

- Funcionalidad implementada y accesible (UI y/o endpoint) conforme a la HU.
- Persistencia validada (base de datos SQL y/o artefactos por caso cuando aplica).
- Validación de entradas y manejo básico de errores (PDF, formularios, consultas).
- Evidencia de prueba ejecutada (capturas, logs o resultados reproducibles).
- Documentación actualizada (README, manual de usuario, notas técnicas o retrospectiva).
- Cambios integrados a Git sin exponer credenciales (uso de variables de entorno y .gitignore).

5.4 Métricas de Jira

Como evidencia cuantitativa de la gestión ágil aplicada en el proyecto AutoSelect-X, se incorporan reportes estándar de Jira correspondientes al tablero AXC. El objetivo de estas métricas no es presentar valores absolutos exhaustivos, sino analizar de manera objetiva los patrones, tendencias y comportamientos que pueden observarse directamente en los gráficos generados por la herramienta. En este sentido, se analizan la velocidad por sprint, la gráfica de control, la gráfica de trabajo hecho y el Diagrama de Flujo Acumulado (CFD), complementados con la trazabilidad de eventos del Sprin.

5.4.1 Velocidad por sprint

El informe de velocidad por sprint permite contrastar visualmente el trabajo confirmado al inicio de cada iteración con el trabajo efectivamente completado al cierre. Tal como se aprecia en la Figura 5.1, existe variabilidad entre sprints en cuanto al volumen de trabajo completado, lo cual es consistente con un proyecto de carácter investigativo y evolutivo, donde el refinamiento del alcance y la maduración de las historias de usuario se realiza de forma progresiva.

En algunos Sprints se observa una menor velocidad de inicio inicial registrada, mientras que en otros la ejecución alcanza o supera el alcance confirmado. Este comportamiento, claramente visible en el gráfico, evidencia un proceso de aprendizaje en la planificación y una adaptación continua del equipo a la complejidad técnica de las actividades abordadas. La métrica de velocidad se utiliza, por tanto, como referencia comparativa entre iteraciones y no como un indicador rígido de desempeño.

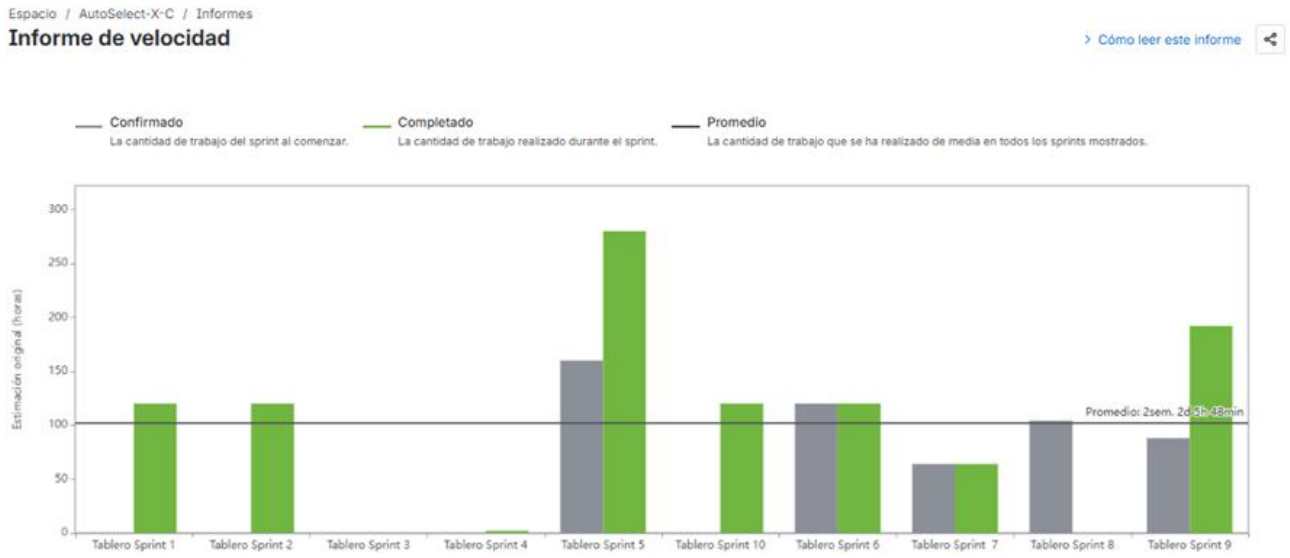


Figura 5.1. Informe de velocidad por sprint (confirmado vs completado).

5.4.2 Gráfica de control

La gráfica de control muestra el tiempo de ciclo de las incidencias a lo largo del tiempo, permitiendo identificar variaciones, dispersión y posibles valores atípicos. En la Figura 5.2 se distinguen claramente el promedio global, el promedio móvil y la banda asociada a la desviación estándar, elementos que facilitan una lectura estadística del comportamiento del flujo de trabajo.

De la inspección visual del gráfico se aprecia un aumento en la dispersión del tiempo de ciclo en los periodos finales, así como la presencia de incidencias con tiempos significativamente mayores al promedio. Este patrón es coherente con fases del proyecto donde se incrementa la

complejidad de integración, validación o resolución de dependencias. La gráfica de control constituye así una herramienta clave para detectar oportunidades de mejora en la gestión del flujo y en la mitigación de bloqueos.

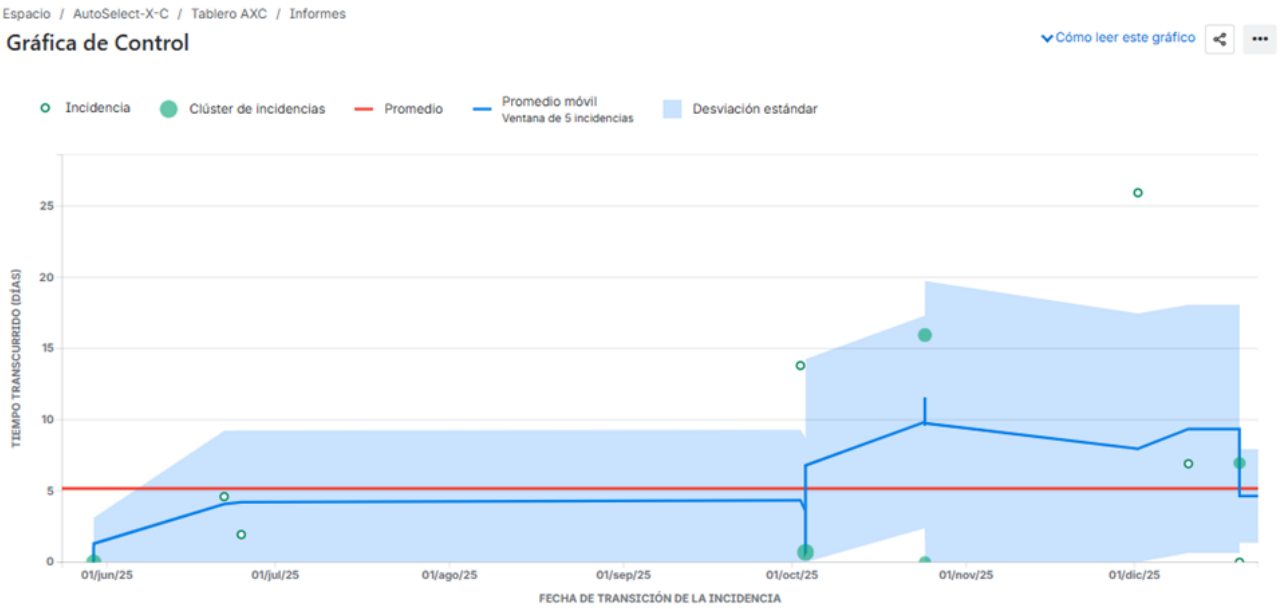


Figura 5.2. Gráfica de control del tiempo de transición por incidencia.

5.4.3 Gráfica de trabajo hecho (work done / burnup)

La gráfica de trabajo hecho, o burnup, permite analizar la relación entre el alcance definido para un sprint y el trabajo completado de forma acumulada. A diferencia del burndown, esta representación facilita observar variaciones en el alcance y la progresión real del trabajo terminado.

En la Figura 5.3 se observa un alcance que permanece estable durante el sprint, mientras que el trabajo completado incrementa hasta alcanzar dicho alcance al final del periodo. Este comportamiento, claramente identificable en la gráfica, indica control del alcance y cierre efectivo de las actividades comprometidas, reforzando la trazabilidad entre planificación y ejecución.

Figura 5.3. Gráfica de trabajo hecho (burnup) del sprint analizado.

5.4.4 Eventos del sprint

La tabla de eventos del sprint proporciona evidencia directa de las fechas de inicio y cierre, así como de la incidencia asociada al trabajo desarrollado. La Figura 5.4 permite verificar la duración efectiva del sprint y la correspondencia entre la historia de usuario trabajada y su estado final.

Este registro refuerza la trazabilidad temporal del proceso ágil, mostrando de manera explícita el ciclo completo de la iteración desde su apertura hasta su cierre, sin necesidad de inferir información adicional no contenida en la herramienta.

Fecha	Tipo de Evento	Incidencia	Trabajo terminado	Alcance del trabajo
3/dic/25 10:40 PM	Sprint iniciado	AXC-10 HU:8 Como usuario quiero que de forma automática se consulte la base de datos técnica, obteniendo sugerencias de bombas compatibles con los requisitos extraídos.	0	1w 3d
19/dic/25 4:47 PM	Incidencia terminada	AXC-10 HU:8 Como usuario quiero que de forma automática se consulte la base de datos técnica, obteniendo sugerencias de bombas compatibles con los requisitos extraídos.	0 → 1w 3d	1w 3d
19/dic/25 4:47 PM	Sprint finalizado	AXC-10 HU:8 Como usuario quiero que de forma automática se consulte la base de datos técnica, obteniendo sugerencias de bombas compatibles con los requisitos extraídos.	1w 3d	1w 3d

Figura 5.4. Registro de eventos del sprint (inicio y cierre).

5.4.5 Diagrama de Flujo Acumulado (CFD)

El Diagrama de Flujo Acumulado (CFD) representa la evolución del número de incidencias en cada estado del flujo de trabajo a lo largo del tiempo. En la Figura 5.5 se distinguen los estados 'Por hacer', 'En curso' y 'Listo', permitiendo evaluar estabilidad del flujo y acumulación por estados.

A partir de la observación del CFD se aprecia que el volumen de trabajo en curso se mantiene relativamente acotado, mientras que los incrementos en el estado 'Listo' se producen de forma escalonada. Este patrón sugiere cierres concentrados tras fases de integración o validación, así como un control razonable del trabajo en progreso (WIP). El CFD constituye, por tanto, una evidencia visual del comportamiento global del sistema de trabajo durante el periodo analizado.

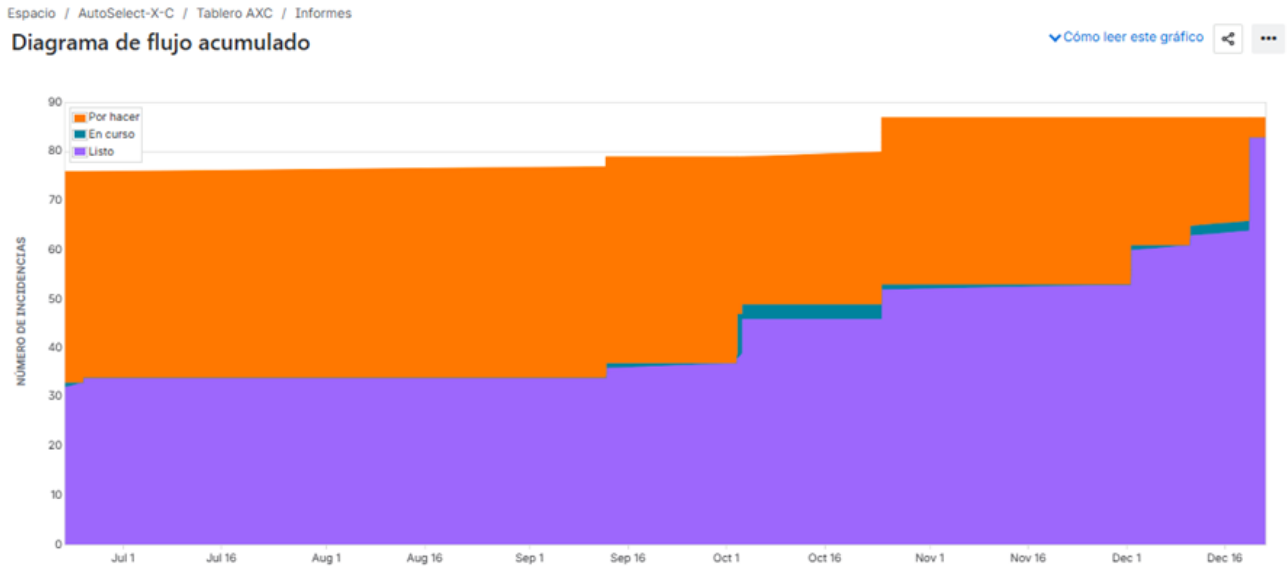


Figura 5.5. Diagrama de Flujo Acumulado (CFD) del tablero AXC.

5.4.6 Evidencias de ceremonias ágiles y colaboración

Complementariamente a las métricas gráficas, la gestión ágil del proyecto se apoyó en ceremonias y prácticas de colaboración, tales como seguimientos periódicos, revisiones de sprint y retrospectivas. Estas actividades se evidencian indirectamente en Jira mediante cambios de estado, comentarios en incidencias y cierre de historias de usuario.

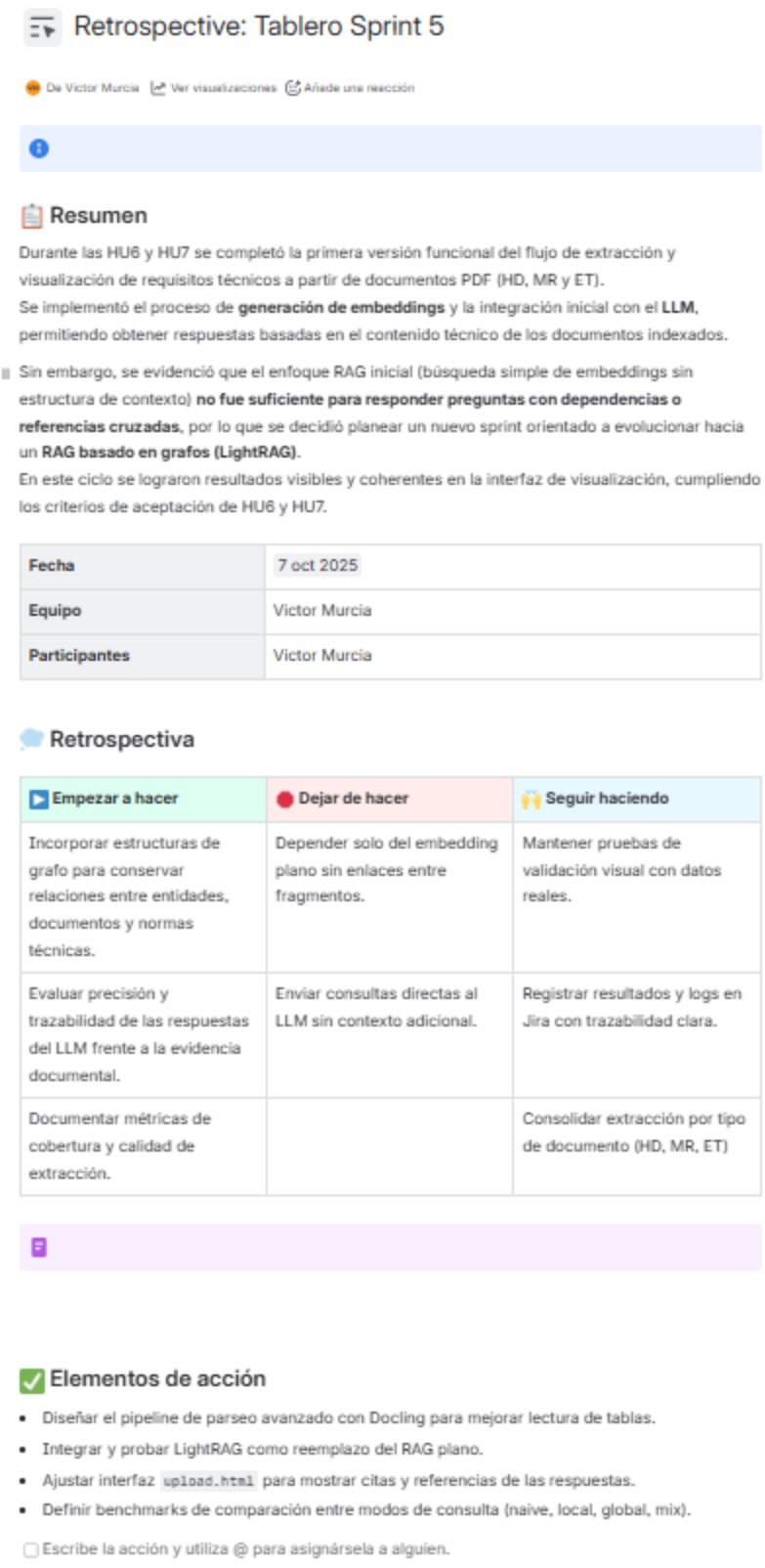
Las retrospectivas realizadas permitieron identificar oportunidades de mejora en la planificación, en la gestión del flujo y en la priorización del trabajo, contribuyendo a una adaptación progresiva del proceso. Aunque su contenido detallado se documenta en anexos o enlaces externos, su impacto es observable en la evolución de las métricas presentadas en esta sección.

5.4.7 Sprint Review y Retrospectivas

Este numeral consolida la evidencia cualitativa asociada a las actividades de Sprint Review y retrospectiva desarrolladas durante el proyecto AutoSelect-X. Dichas prácticas complementan las métricas cuantitativas de Jira, permitiendo evaluar el incremento entregado, reflexionar sobre el proceso y definir acciones de mejora.

Ejemplo representativo: Sprint 5

El Sprint 5 se selecciona como ejemplo representativo debido a la claridad y profundidad de su retrospectiva. Durante este sprint se consolidó la primera versión funcional del flujo de extracción de requisitos desde PDFs y se evidenció la necesidad de evolucionar el enfoque RAG inicial hacia una arquitectura basada en grafos (LightRAG).



Retrospective: Tablero Sprint 5

De Victor Murcia Ver visualizaciones Añade una reacción

📋 Resumen

Durante las HU6 y HU7 se completó la primera versión funcional del flujo de extracción y visualización de requisitos técnicos a partir de documentos PDF (HD, MR y ET).
Se implementó el proceso de **generación de embeddings** y la integración inicial con el **LLM**, permitiendo obtener respuestas basadas en el contenido técnico de los documentos indexados.

Sin embargo, se evidenció que el enfoque RAG inicial (búsqueda simple de embeddings sin estructura de contexto) **no fue suficiente para responder preguntas con dependencias o referencias cruzadas**, por lo que se decidió planear un nuevo sprint orientado a evolucionar hacia un **RAG basado en grafos (LightRAG)**.
En este ciclo se lograron resultados visibles y coherentes en la interfaz de visualización, cumpliendo los criterios de aceptación de HU6 y HU7.

Fecha	7 oct 2025
Equipo	Victor Murcia
Participantes	Victor Murcia

💭 Retrospectiva

▶️ Empezar a hacer	🛑 Dejar de hacer	🙌 Seguir haciendo
Incorporar estructuras de grafo para conservar relaciones entre entidades, documentos y normas técnicas.	Depender solo del embedding plano sin enlaces entre fragmentos.	Mantener pruebas de validación visual con datos reales.
Evaluar precisión y trazabilidad de las respuestas del LLM frente a la evidencia documental.	Enviar consultas directas al LLM sin contexto adicional.	Registrar resultados y logs en Jira con trazabilidad clara.
Documentar métricas de cobertura y calidad de extracción.		Consolidar extracción por tipo de documento (HD, MR, ET)

✅ Elementos de acción

- Diseñar el pipeline de parseo avanzado con Docling para mejorar lectura de tablas.
- Integrar y probar LightRAG como reemplazo del RAG plano.
- Ajustar interfaz `upload.html` para mostrar citas y referencias de las respuestas.
- Definir benchmarks de comparación entre modos de consulta (naive, local, global, mix).

Escribe la acción y utiliza @ para asignársela a alguien.

Figura 5.6. Ejemplo de retrospectiva documentada en Confluence (Sprint 5).

Resumen de Sprint Reviews documentadas

La Tabla 5.4 resume de manera cualitativa los principales focos y aprendizajes derivados de las retrospectivas documentadas en los sprints iniciales, evitando valores no parametrizados y limitándose a información verificable.

Sprint	Foco del Sprint	Aprendizaje principal	Impacto posterior
Sprint 1	Gestión de usuarios	Necesidad de estimación previa y validación de cierres	Mejor planificación
Sprint 2	Seguridad y carga de PDFs	Importancia de validaciones tempranas	Base para procesamiento documental
Sprint 3	Pipeline y visualización	Limitaciones de estimación por tiempo	Uso de History Points
Sprint 4	Uso inicial de LLM	Relevancia del contexto técnico	Prompts más controlados
Sprint 5	Flujo RAG inicial	Insuficiencia del RAG plano	Migración a LightRAG
Sprint 6	Refinamiento backend	Necesidad de documentación técnica	Mejor mantenibilidad

Tabla 5.4. Resumen cualitativo de Sprint Reviews y retrospectivas.

En sprints posteriores, las reflexiones se realizaron de forma menos estructurada o exploratoria, por lo cual no se incluyen como retrospectivas formales en este análisis. No obstante, las decisiones tomadas en los sprints iniciales influyeron directamente en la arquitectura, el flujo de trabajo y la estabilidad alcanzada en las iteraciones finales del proyecto.

5.5 Roles, responsabilidades y colaboración

En el desarrollo del proyecto AutoSelect-X, el autor asumió de manera integrada los roles de Product Owner, Scrum Master y Developer, una configuración habitual en proyectos académicos individuales de carácter investigativo y experimental. Esta convergencia de roles permitió una alineación directa entre la visión del producto, la toma de decisiones técnicas y la ejecución del desarrollo, reduciendo fricciones de coordinación y facilitando ciclos de retroalimentación más cortos.

Desde la perspectiva de Product Owner, el autor fue responsable de la definición y priorización del backlog, orientando las historias de usuario según su valor funcional, su impacto en los objetivos del proyecto y el riesgo técnico asociado. Esta priorización se realizó de forma iterativa, ajustándose a los hallazgos obtenidos en cada sprint, especialmente aquellos derivados de pruebas con documentos reales, integración de modelos de lenguaje y evolución del enfoque RAG.

En el rol de Scrum Master, se mantuvo la disciplina del proceso ágil mediante el uso sistemático de Jira como herramienta central de gestión. Esto incluyó la visualización del flujo de trabajo, el seguimiento del avance por sprint, la revisión del estado de las incidencias y la identificación temprana de bloqueos. La adopción de métricas como velocidad, gráficas de control y diagramas de flujo acumulado permitió monitorear el proceso y apoyar decisiones de mejora continua, aun en un contexto de equipo unipersonal.

Finalmente, como Developer, el autor ejecutó el desarrollo end-to-end del sistema, abarcando desde la implementación del backend, el procesamiento de documentos PDF y la integración de modelos de lenguaje, hasta la construcción de prototipos funcionales y pruebas técnicas. Esta responsabilidad integral favoreció una comprensión profunda del dominio del problema y de las implicaciones técnicas de cada decisión de diseño.

La colaboración y retroalimentación externa desempeñaron un papel clave en la maduración del proyecto. En particular, el acompañamiento del tutor y del espacio académico se constituyó en un mecanismo fundamental de validación conceptual y metodológica. Las observaciones, recomendaciones y lineamientos recibidos durante las socializaciones y entregas parciales se incorporaron de manera explícita al backlog y dieron lugar a ajustes tanto en el alcance como en la arquitectura técnica del sistema.

Asimismo, los Trabajos Integradores 2 y 3 aportaron un marco estructurado de evaluación progresiva que permitió contrastar los avances del proyecto con los objetivos formativos del programa. Estos hitos académicos facilitaron la reflexión crítica sobre las decisiones tomadas, promovieron la formalización de resultados intermedios y fortalecieron la coherencia entre el desarrollo práctico y el sustento teórico del trabajo de grado.

En conjunto, esta combinación de roles integrados, apoyo académico continuo y uso disciplinado de prácticas ágiles permitió conducir el proyecto de manera controlada, trazable y alineada con los objetivos de investigación, maximizando el valor del acompañamiento docente y consolidando un proceso de desarrollo riguroso y reflexivo.

6. Pruebas y Validación del Sistema

Objetivo: garantizar la calidad del producto entregado mediante un proceso de pruebas y validación incremental, centrado en el flujo completo de AutoSelect-X (carga de documentos, extracción, normalización, consulta asistida por RAG y visualización/exportación). Dado que el sistema trabaja con PDFs reales de ingeniería y combina heurísticas, extracción estructurada y un componente RAG, la estrategia de validación prioriza: (I) robustez del procesamiento documental, (II) consistencia de normalización en unidades estándar, (III) trazabilidad hacia la fuente (página/fragmento) y (IV) estabilidad del comportamiento ante cambios de configuración o prompts.

6.1 Plan de pruebas (unitarias, de integración, funcionales y de aceptación)

El plan de pruebas se definió para cubrir componentes críticos y el flujo end-to-end, combinando pruebas unitarias con pruebas de integración y pruebas funcionales. Adicionalmente, se definieron pruebas de aceptación por inspección técnica, comparando salidas contra 'ground truth' visible en los documentos (p. ej., cantidad de bombas, tags y parámetros de operación), evitando inferencias no sustentadas.

Tipos de prueba considerados:

- Unitarias: validación del normalizador (parseo numérico, conversión de unidades, control de nulos) y reglas/guardarraíles (p. ej., no inventar valores).
- Integración: ejecución por caso (carga de PDF → extracción → normalización → persistencia/consulta), verificando continuidad del pipeline.
- Funcionales: validación de funcionalidades visibles en la aplicación (login, creación de caso, carga de PDFs, consulta asistida, exportación/descarga).
- Aceptación: verificación técnica comparando resultados con evidencias del documento (ground truth) y revisando consistencia y trazabilidad.
- Regresión: re-ejecución de casos de referencia cuando se ajustan prompts/heurísticas/configuración del RAG, para detectar degradaciones.

6.2 Cobertura de pruebas

La cobertura se expresa en términos de cobertura funcional y cobertura de extracción/normalización. En particular, para el dominio de selección de bombas, la cobertura se analiza con métricas operacionales que pueden calcularse sobre casos reales: recall de bombas (detección de equipos esperados), precisión (control de falsos positivos), completitud de campos por TAG y porcentaje de normalización exitosa sobre valores efectivamente extraídos.

Métrica	Definición operacional	Interpretación en el proyecto
Recall de bombas	Bombas detectadas / bombas esperadas (ground truth)	Cobertura de extracción de equipos.
Precisión	1 − (Falsos positivos / detecciones totales)	Control de ruido: evitar confundir códigos con TAGs.
Completitud por TAG	% de campos mínimos con valor (RAW vs NORMALIZED)	Utilidad para selección técnica; el normalizador deja nulos si no hay evidencia.
% Normalización	Conversiones exitosas / valores con dato en RAW	Capacidad de estandarizar unidades sin inventar información.
Trazabilidad	% de campos con referencia a fuente	Auditoría técnica: evidencia de página/fragmento.

Tabla 6.1. Métricas de cobertura y evaluación para extracción y normalización.

6.3 Matriz de trazabilidad entre requisitos y pruebas

La trazabilidad asegura que cada requisito (HU/criterio de aceptación) esté cubierto por uno o más casos de prueba con evidencia verificable. En sistemas RAG, la trazabilidad incluye no solo la ejecución del flujo, sino también la capacidad de justificar resultados con referencia documental.

Requisito / HU (síntesis)	**Tipo de prueba**	**Caso de prueba**	**Criterio de aprobación**	**Evidencia**
Carga segura de PDFs y creación de caso	Funcional	TC-FUN-01	Usuario puede crear caso y cargar PDFs sin errores	[Captura/UI]
Extracción de bombas y TAGs	Integración/Aceptación	TC-INT-01	Lista de bombas coincide con ground truth (cuando aplica)	[Resultados/Anexo]
Normalización de unidades	Unitarias + Integración	TC-UNI-01 / TC-INT-02	Conversión a unidades estándar; sin inventar valores	[Logs/JSON]
Aislamiento por caso (sin contaminación)	Regresión	TC-REG-01	Respuestas limitadas al caso activo	[Ejecución benchmark]
Exportación/descarga de resultados	Funcional	TC-FUN-02	Exportación consistente con datos en UI	[Archivo exportado]

Tabla 6.2. Matriz de trazabilidad (plantilla base; completar evidencias en anexos).

6.4 Resultados de las pruebas

Los resultados se documentan a partir de ejecuciones end-to-end sobre casos reales. Para efectos de evidencia en este trabajo final, se incorporan tres casos de prueba presentados en la sustentación del Proyecto Integrador 3, donde se reportan métricas comparables contra 'ground truth' explícito en los PDFs.

Resumen de resultados cuantitativos (casos de prueba):

Caso	**Bombas esperadas**	**Bombas detectadas**	**Recall**	**Precisión / FP**	**Completitud y normalización (según evidencia)**
Caso 38 (PV008409-24)	6	6	100%	FP = 0 (precisión 100%)	Completitud y % normalización reportados en la evidencia del caso.
Caso 33 (PV009257_25)	24	20	83.3%	Se reportan TAGs detectados; revisión FP por tablas	Completitud RAW ≈ 73.3%; NORMALIZED ≈ 61.1%; % normalización ≈ 69.7% (53/76).
PV008977_25	1	4 (1 correcta)	100% (1/1)	FP = 3 (códigos confundidos como TAG)	Completitud RAW 87.5%; NORMALIZED 79.2%; evidencia de necesidad de filtrar TAGs documentales.

Tabla 6.3. Resumen de resultados por caso.

Interpretación de resultados (con base en la evidencia): (I) el Caso 38 muestra desempeño consistente cuando el documento contiene un conjunto de bombas claramente definido (cantidad y TAGs explícitos), (II) el Caso 33 evidencia retos de cobertura en documentos con mayor volumen y tablas extensas (detección parcial), y (III) el caso PV008977_25 evidencia un escenario crítico de falsos positivos por confusión entre códigos de documento y TAGs de equipo, lo que justifica la necesidad de reglas de filtrado y/o heurísticas específicas para diferenciar identificadores documentales de etiquetas de activos.

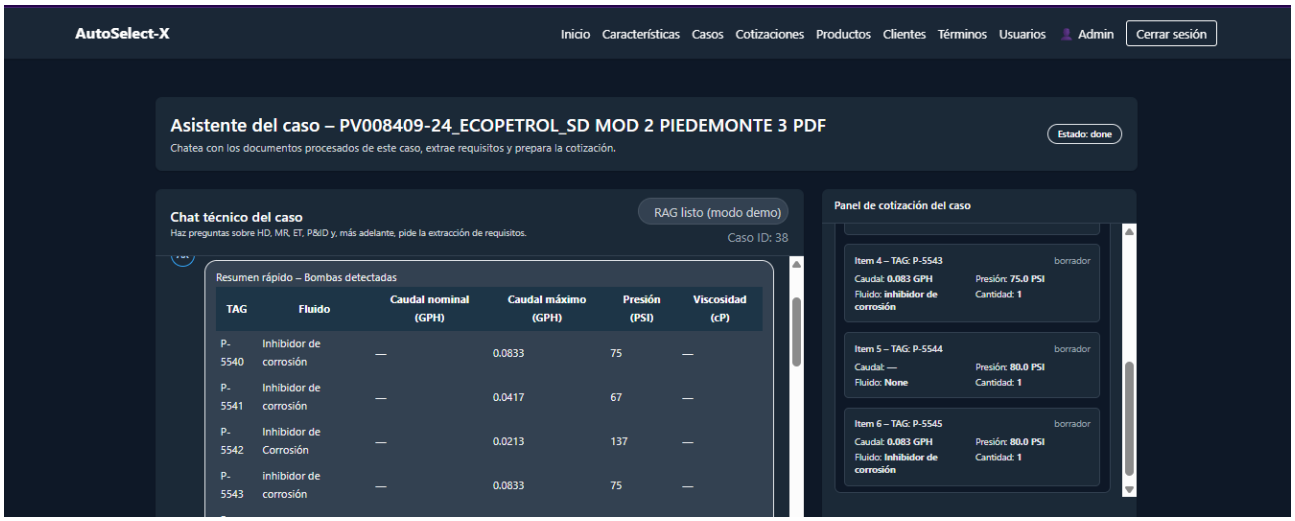


Figura 6.1. Evidencia de resultados – Caso 38.

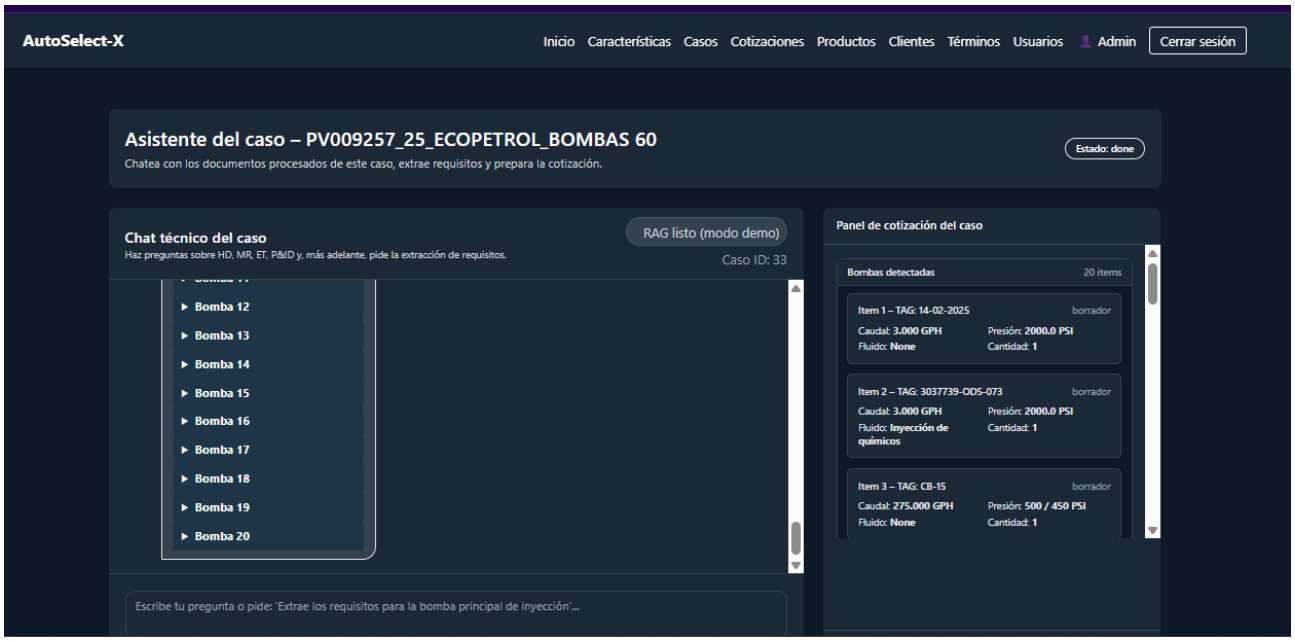


Figura 6.2. Evidencia de resultados – Caso 33.

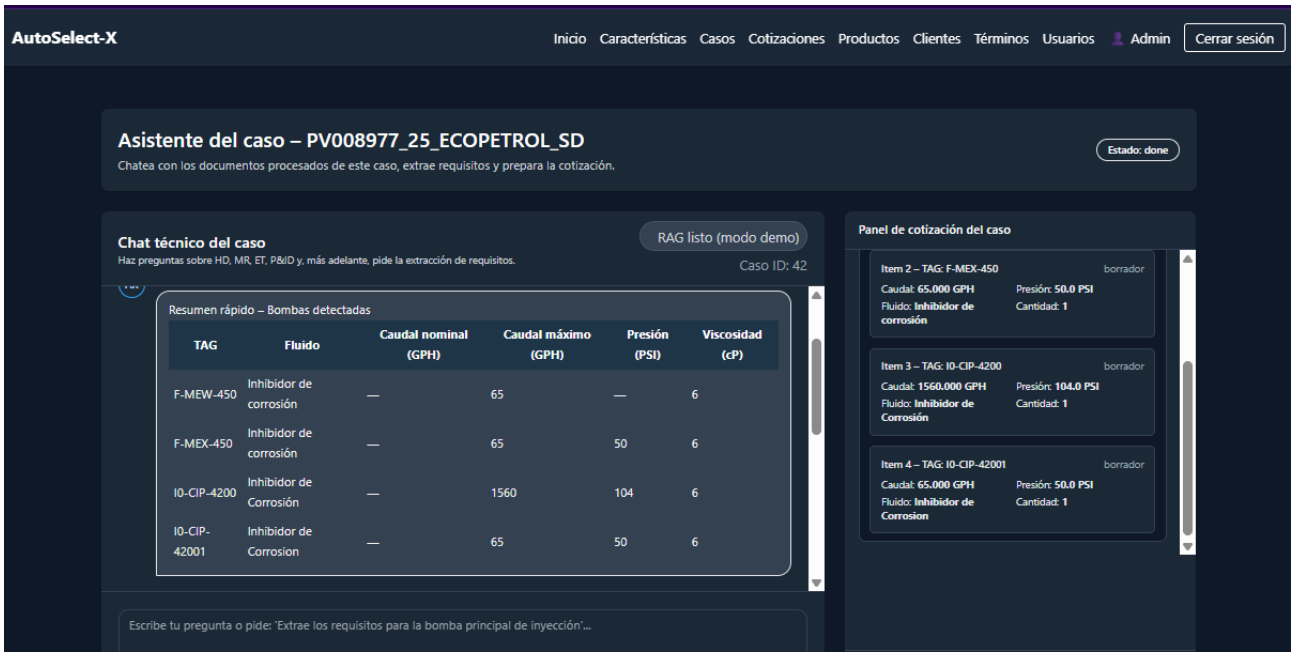


Figura 6.3. Evidencia de resultados – Caso PV008977_25.

Tabla 6.4. Métricas principales de evaluación del sistema.

Métrica	**Definición**	**Propósito**
Recall de bombas	Bombas detectadas / bombas esperadas (referencia manual)	Medir cobertura de extracción.
Completitud por TAG	% bombas con TAG + parámetros mínimos (flow y pressure)	Medir utilidad para selección.
% Normalización	% campos convertidos a unidades estándar sin nulos	Medir consistencia para comparación.
Precisión (FP)	Bombas extraídas que no corresponden a bombas reales	Controlar ruido en resultados.
Trazabilidad	% campos con referencia a fuente (página/fragmento)	Soportar auditoría técnica.

6.5 Pruebas con usuarios (si aplica)

Debido al carácter académico del proyecto, a su duración acotada y al enfoque prioritario en la validación técnica del pipeline de extracción y selección, no fue posible realizar pruebas formales con usuarios externos durante el periodo de ejecución del trabajo. En consecuencia, no se desarrollaron estudios de usabilidad ni sesiones estructuradas de evaluación con múltiples perfiles de usuario final.

No obstante, el sistema fue utilizado y evaluado de manera continua por el propio autor, quien actuó como usuario técnico de referencia, ejecutando de forma reiterada el flujo completo de la aplicación en diferentes escenarios y casos reales. Esta validación funcional dirigida permitió verificar la correcta navegabilidad, la coherencia del flujo de interacción y la estabilidad operativa del sistema en condiciones de uso realista.

En particular, se validaron de manera sistemática las siguientes actividades como usuario de la aplicación:

- Autenticación y acceso seguro al sistema.
- Creación y gestión de casos de análisis.
- Carga y procesamiento de documentos técnicos en formato PDF (HD, ET y MR).
- Ejecución de consultas de extracción y selección de bombas.
- Revisión de resultados estructurados, incluyendo parámetros técnicos y referencias de trazabilidad hacia el documento fuente.
- Exportación de resultados y verificación de consistencia entre la visualización y los artefactos generados.

Esta autoevaluación funcional permitió detectar y corregir problemas de navegación, organización de la información, claridad en la presentación de resultados y manejo de valores faltantes, contribuyendo a la mejora progresiva de la experiencia de uso del prototipo.

Finalmente, se reconoce que la realización de pruebas con usuarios externos, incluyendo perfiles del sector de ingeniería o ventas técnicas, constituye una línea de trabajo futura relevante. Dichas pruebas permitirían evaluar de forma más exhaustiva la usabilidad, la curva de aprendizaje y la utilidad percibida del sistema en un contexto operativo real, y se proponen como parte de una fase posterior de maduración del producto.

7. Despliegue y Manuales

Objetivo: asegurar la disponibilidad y el uso adecuado de la aplicación AutoSelect-X mediante la definición de ambientes, procedimientos de despliegue y manuales operativos. El despliegue del prototipo se realizó sobre la plataforma Render, con base de datos PostgreSQL administrada y configuración por variables de entorno. La aplicación se publica en: https://autoselectxv1.onrender.com/.

7.1 Ambientes de despliegue (desarrollo, pruebas y producción)

Se definieron tres ambientes conceptuales para soportar el ciclo de vida del prototipo:

Ambiente	Propósito	Características	Evidencia/artefactos
Desarrollo (local)	Implementación y depuración	Ejecución con venv, logs detallados, pruebas rápidas por scripts	Ejecución local con Uvicorn + scripts de soporte
Pruebas (staging conceptual)	Validación de integración antes de publicar	Re-ejecución de pipeline, seed y limpieza de cache; verificación de UI/JSON	Correr scripts en consola (local o shell) y validar casos
Producción (Render)	Disponibilidad del prototipo	Despliegue web, variables de entorno, DB PostgreSQL, start command	URL pública + consola Shell de Render para mantenimiento

Tabla 7.1. Ambientes de despliegue definidos para AutoSelect-X.

7.2 Manual de instalación y configuración

Este manual describe los pasos mínimos para instalar y configurar la aplicación en entorno local y para su despliegue en Render. Por seguridad, los valores sensibles (por ejemplo, llaves API) no se documentan como texto plano; en su lugar, se especifican las variables requeridas y su finalidad.

7.2.1 Requisitos (entorno local)

Requisitos mínimos recomendados: Python 3.10+; acceso a PostgreSQL (local o remoto); Git; y variables de entorno configuradas.

Instalación en local (ejemplo):

```
python -m venv venv
source venv/bin/activate  # Linux/macOS
# o .\venv\Scripts\activate  # Windows
```

pip install -r requirements.txt
uvicorn app.main:app --host 127.0.0.1 --port 8080

7.2.2 Variables de entorno

La aplicación utiliza variables de entorno para parametrizar conexión a base de datos, credenciales externas y rutas internas. En Render estas variables se definen desde el panel de configuración del servicio (Environment Variables). Se recomienda almacenar secretos en el gestor de variables y no versionarlos en el repositorio.

Variable	Descripción	Ejemplo (sin secretos)
DATABASE_URL / DATABASE_URI	Cadena de conexión a PostgreSQL	postgresql://USER:***@HOST:PORT/DBNAME
OPENAI_API_KEY	Llave para consumo del modelo LLM (si aplica)	sk-*** (definida solo en entorno)
PYTHONPATH	Ruta base del proyecto en Render	/opt/render/project/src
PROJECT_ROOT	Ruta raíz del proyecto (si aplica)	/opt/render/project/src

Tabla 7.2. Variables de entorno requeridas (valores sensibles omitidos).

7.2.3 Inicialización de base de datos y datos semilla

Una vez configurada la base de datos en Render, se ejecutaron scripts de inicialización desde la consola Shell de Render para crear tablas, cargar datos requeridos por el selector y habilitar un usuario administrador inicial. Los scripts se ejecutan con el mismo entorno del servicio (mismas variables de entorno), lo cual reduce inconsistencias entre local y producción.

Comandos de mantenimiento ejecutables desde la Shell (ejemplos):

```
# Crear todas las tablas definidas por los modelos (SQLAlchemy)
python -m app.scripts.create_all_tables


# Crear/actualizar usuario administrador inicial
python -m app.scripts.create_admin_user

# Cargar catálogos/tablas del selector (Milton Roy)
python -m app.scripts.load_mroy_catalog

# Cargar valores por defecto (catálogos base)
python -m app.scripts.seed_defaults

# Limpiar cache/almacenamiento RAG (modo seguro con dry-run)
python -m app.scripts.reset_rag_cache --all --wipe-shared --dry-run
```

```
# Ejecutar borrado real (si procede)
python -m app.scripts.reset_rag_cache --all --wipe-shared
```

Listado 7.1. Comandos de inicialización y mantenimiento ejecutables en Shell (Render/local).

Nota: el script 'seed_defaults.py' se utiliza para crear registros mínimos requeridos por la aplicación (por ejemplo, un 'Customer' activo y un 'DeliveryTerm' activo) cuando no existen en la base de datos. Esto garantiza que ciertos formularios o flujos dependientes de catálogos no fallen por ausencia de datos.

7.2.4 Despliegue en Render (resumen operativo)

En Render, el despliegue se realizó configurando el repositorio fuente, instalando dependencias y definiendo el comando de arranque del servicio. Un patrón típico de arranque es:

```
uvicorn app.main:app --host 0.0.0.0 --port $PORT
```

Listado 7.2. Comando de inicio del servicio web en Render.

7.3 Manual de usuario

El manual de usuario describe el flujo principal de operación de AutoSelect-X desde la perspectiva de un usuario técnico. Las acciones se realizan desde la interfaz web publicada (Render), y el usuario trabaja sobre 'casos' para agrupar documentos y resultados.

Paso	Descripción
1. Acceso	Ingresar a la URL de la aplicación y autenticarse con credenciales válidas.
2. Crear caso	Crear un caso (identificador y nombre) para agrupar los documentos del análisis.
3. Cargar PDFs	Subir documentos técnicos (HD/ET/MR) asociados al caso.
4. Procesar	Ejecutar el procesamiento/extracción para generar resultados estructurados.
5. Consultar	Realizar consultas guiadas (asistente) y revisar respuestas y evidencias asociadas.
6. Exportar	Exportar resultados (p. ej., JSON) y descargar artefactos del caso.

Tabla 7.3. Flujo principal del usuario en AutoSelect-X.

7.4 Manual técnico y de mantenimiento

El mantenimiento del prototipo se concentra en operaciones de (I) base de datos y catálogos, (II) limpieza de almacenamiento/caches asociados al RAG por caso y (III) verificación de arranque y logs. Estas actividades se ejecutan desde la Shell de Render o en entorno local con las mismas variables de entorno.

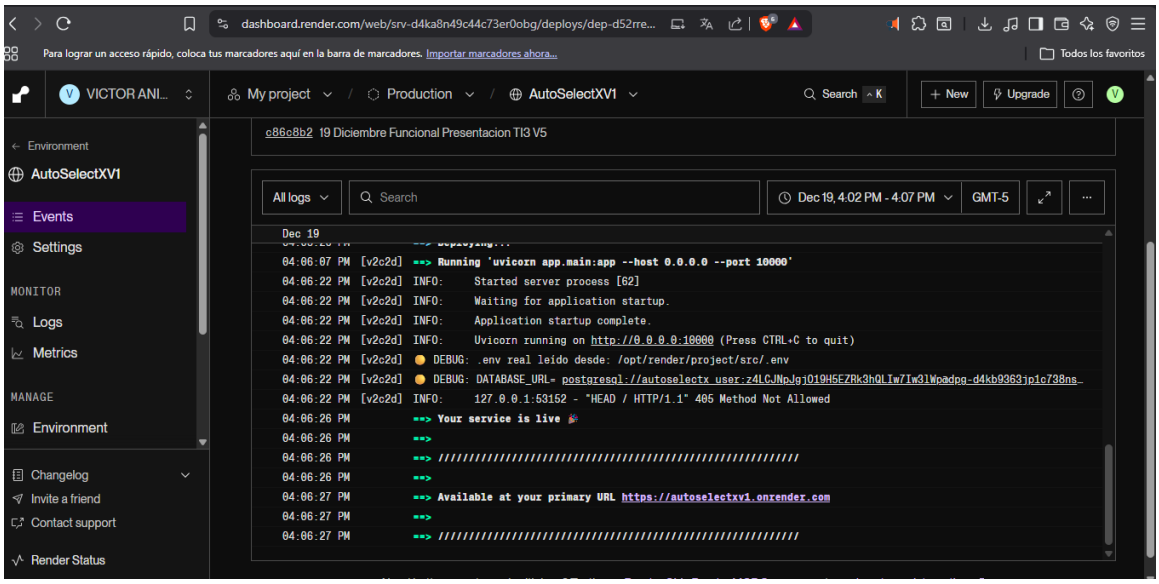
Tarea	**Script/Comando**	**Cuándo ejecutar**
Crear todas las tablas	python -m app.scripts.create_all_tables	Primer despliegue o recuperación de acceso
Crear/activar admin	python -m app.scripts.create_admin_user	Primer despliegue o recuperación de acceso
Cargar catálogos	python -m app.scripts.load_mroy_catalog	Actualización de catálogos/listas de selección
Semillas mínimas	python -m app.scripts.seed_defaults	Tras DB nueva o limpieza de registros base
Reset cache RAG	python -m app.scripts.reset_rag_cache --all --wipe-shared	Antes de pruebas de regresión o ante inconsistencias
Selección (diagnóstico)	python -m app.scripts.select_mroy_pump	Validación técnica de reglas/selección desde DB

Tabla 7.4. Operaciones de mantenimiento y scripts asociados.

Adicionalmente, Render permite observar logs del servicio, reiniciar despliegues y actualizar variables de entorno. Ante cambios de dependencias, se recomienda reconstruir la imagen del servicio (redeploy) y verificar el arranque del servidor y la conectividad con la DB.

7.5 Evidencia de despliegue

Como evidencia de despliegue, se recomienda anexar capturas del panel de Render (servicio activo), la configuración de variables de entorno (sin secretos) y la ejecución de scripts en la Shell. Alternativamente, puede anexarse un video corto demostrativo mostrando: acceso, creación de caso, carga de un PDF, ejecución de extracción y exportación de resultados.



Espacio reservado: Figura 7.5. Evidencia del panel de Render.

8. Lecciones Aprendidas y Mejora Continua

Objetivo: fomentar la reflexión crítica sobre el proceso de construcción del prototipo AutoSelect-X y consolidar aprendizajes técnicos y organizativos que orienten mejoras futuras. Dado que el proyecto integró procesamiento documental y un componente RAG, se identificaron riesgos específicos de trazabilidad, consistencia de extracción y estabilidad operativa

8.1 Dificultades técnicas y organizativas

Las principales dificultades se relacionaron con (I) variabilidad de los PDFs (tablas, formatos y códigos), (II) ambigüedad entre identificadores de documento y TAGs de equipo, (III) manejo de caches/almacenamiento del RAG por caso y (IV) coordinación de cambios entre pipeline y capa de aplicación.

8.2 Cambios en la planificación

Durante el desarrollo se realizaron ajustes de planificación para controlar riesgos. En particular, se priorizó asegurar el procesamiento y normalización (base determinística) antes de ampliar funcionalidades asistidas por LLM/RAG. Asimismo, se adoptó una estrategia de pruebas por casos de referencia y se incorporaron scripts de mantenimiento para facilitar regresión.

8.3 Fortalezas y debilidades del equipo

Al tratarse de un proyecto ejecutado principalmente por un solo desarrollador (autor), la principal fortaleza fue la coherencia end-to-end y la trazabilidad de decisiones técnicas. Como debilidad, la capacidad de paralelización fue limitada, afectando la posibilidad de realizar pruebas con usuarios externos y documentación adicional.

8.4 Proyecciones de mejora del sistema

Como mejoras prioritarias se proponen: (I) fortalecer el filtrado de TAGs para reducir falsos positivos en documentos con códigos similares, (II) ampliar el conjunto de casos de regresión, (III) incrementar la trazabilidad automática por campo (referencia de página/tabla), (IV) mejorar observabilidad (logs/métricas) y (V) ejecutar pruebas formales con usuarios del perfil objetivo.

8.5 Aprendizajes personales y profesionales

El proyecto consolidó competencias en ingeniería de software aplicada a IA: diseño incremental, gestión de riesgos, despliegue en la nube y validación orientada a evidencia. En el plano técnico, se fortaleció el criterio para diferenciar componentes determinísticos (extracción/normalización) de componentes probabilísticos (LLM/RAG), estableciendo etapas para evitar inferencias sin respaldo.

Hallazgo	**Impacto observado**	**Acción aplicada**	**Mejora futura**
Falsos positivos en TAGs	Códigos documentales detectados como bombas	Análisis por casos y reglas de depuración	Heurísticas/regex por dominio + validación cruzada con contexto
Cobertura variable en tablas extensas	Detección parcial en documentos grandes	Casos de prueba dedicados y revisión de extracción	Mejoras de parsing y segmentación; refinamiento de tablas
Inconsistencias por cache RAG	Resultados cruzados o desactualizados	Reset controlado de cache y aislamiento por caso	Política de invalidación automática y trazabilidad por caso
Despliegue y operaciones	Necesidad de scripts para DB/seed	Scripts ejecutables en Shell Render	Automatización segura (jobs) + monitoreo

Tabla 8.1. Lecciones aprendidas y acciones de mejora continua.

9. Anexos

Objetivo: incluir elementos complementarios relevantes para facilitar verificación, replicación y auditoría del proyecto. Los anexos propuestos a continuación deben completarse con los enlaces y artefactos definitivos correspondientes al repositorio, datasets y presentación final.

9.1 Enlaces a repositorios de código

Repositorio principal del proyecto:

REPO: https://github.com/VmurciaU/AutoSelectXV1

9.2 Referencias bibliográficas o técnicas

- RAG (Retrieval-Augmented Generation) – paper base (arXiv:2005.11401): https://arxiv.org/abs/2005.11401
- FastAPI – documentación oficial: https://fastapi.tiangolo.com/
- PostgreSQL – documentación oficial: https://www.postgresql.org/docs/
- Render – documentación oficial (despliegue): https://render.com/docs
- pdfplumber – repositorio/documentación: https://github.com/jsvine/pdfplumber
- LightRAG – repositorio/documentación: https://github.com/HKUDS/LightRAG

9.3 Presentación final

Se anexa la presentación final. Además, se incluye el enlace a la aplicación desplegada.

APP (Render):
https://autoselectxv1.onrender.com/

PRESENTACIÓN:
https://docs.google.com/presentation/d/1h1SXjJBWG2vmzsaz7ghkvAYCkvipcy1g2mooYcWaHps/edit?usp=sharing