UNMdP - Facultad de Ingeniería - Departamento de Informática

LawCEDIC: Plataforma web para la gestión de documentos judiciales vinculados a los ciberdelitos y la evidencia digital para el Observatorio de Cibercrimen y Evidencia Digital en Investigaciones Criminales (OCEDIC) de la Facultad de Derecho de la Universidad Austral

Autores

- Avalos, Wenceslao
- Lapiana, Santiago Nicolás
- Sosa, Santiago Fabián

Director

- Hinojal, Hernán

Proyecto final para optar por el grado de Ingeniero en Informática

Mar del Plata, 19 de marzo de 2025

UNMdP - Facultad de Ingeniería - Departamento de Informática

LawCEDIC: Plataforma web para la gestión de documentos judiciales vinculados a los ciberdelitos y la evidencia digital para el Observatorio de Cibercrimen y Evidencia Digital en Investigaciones Criminales (OCEDIC) de la Facultad de Derecho de la Universidad Austral

Autores

- Avalos, Wenceslao
- Lapiana, Santiago Nicolás
- Sosa, Santiago Fabián

Director

- Hinojal, Hernán

Proyecto final para optar por el grado de Ingeniero en Informática

Mar del Plata, 19 de marzo de 2025

Agradecimientos

A nuestras familias y amigos por su apoyo incondicional.

A los docentes de la Facultad de Ingeniería, que nos brindaron las herramientas y conocimientos para poder llegar a esta instancia de evaluación.

A nuestro director de Trabajo Final, Hernán Hinojal, por estar siempre a disposición.

A OCEDIC, particularmente a su directora Daniela Dupuy y el referente funcional Javier Vellido, por darnos la oportunidad de participar en un proyecto de esta envergadura.

A todos ellos, muchas gracias.

Índice

1. Resumen

La plataforma LawCEDIC fue concebida en OCEDIC (Observatorio de Cibercrimen y Evidencia Digital en Investigaciones Criminales) de la Universidad Austral, a cargo de la fiscal Daniela Dupuy. El objetivo principal es optimizar la carga, gestión y búsqueda de documentos judiciales para simplificar las tareas de analistas, abogados, peritos e integrantes del Poder Judicial.

Para su desarrollo, se utilizó una técnica inspirada en las metodologías ágiles, que incluyó la realización de reuniones semanales con el líder del proyecto. Durante estas reuniones, se recibió *feedback* constante, lo que permitió ajustar y mejorar las funcionalidades según las necesidades y prioridades identificadas. El proyecto incluyó la integración de un modelo de inteligencia artificial, que permitió automatizar la extracción de información en el proceso de carga de documentos y habilitó búsquedas mediante lenguaje natural para facilitar el acceso a la información. El despliegue del sistema se llevó a cabo en una máquina virtual configurada con múltiples medidas de seguridad, debido a los objetivos de negocio del demandante, relacionados a la comercialización de este servicio.

Los resultados, obtenidos en casos judiciales de prueba, indicaron que la plataforma cumplió con los objetivos propuestos y demostró un impacto positivo en la eficiencia del manejo de la información relacionada. En conclusión, LawCEDIC representa una solución versátil para el ámbito judicial, ideal para optimizar la administración de documentos judiciales digitalizados. La plataforma es actualmente accesible en https://lawcedic.ocedic.com y continúa en proceso de perfeccionamiento.

2. Introducción

En la era digital, el acceso a la información se ha convertido en un elemento esencial en todos los ámbitos de la sociedad. La cantidad de datos disponibles crece de manera exponencial, lo que ha generado nuevos desafíos relacionados con la búsqueda, organización y análisis eficiente de grandes volúmenes de contenido. Si bien la disponibilidad de información es crucial, el verdadero valor reside en contar con herramientas capaces de filtrar, identificar y extraer los datos más relevantes de forma precisa y oportuna. Este desafío es particularmente significativo en el ámbito judicial, donde la gestión de documentos como leyes, jurisprudencia y doctrinas requiere de un alto grado de exactitud y rapidez.

En este contexto, la inteligencia artificial (IA) ha emergido como una solución innovadora que permite abordar las complejidades asociadas con la gestión de información. En los últimos años, los grandes modelos lingüísticos (LLM, por sus siglas en inglés) han revolucionado la manera en que se procesan y analizan datos textuales. Estos modelos, diseñados para interpretar y generar texto en lenguaje natural, tienen la capacidad de simplificar el acceso a información clave, reduciendo la carga administrativa y agilizando el tiempo dedicado a tareas repetitivas. En el ámbito judicial, esta capacidad resulta valiosa al permitir que los profesionales del derecho accedan a documentos relevantes de manera más eficiente.

Es en este marco que surge LawCEDIC. Su objetivo es centralizar, organizar y facilitar el acceso a documentos judiciales relacionados a las cuatro temáticas con las que trabaja el observatorio: ciberacosos a niños, niñas y adolescentes; violencia de género en las TIC; Inteligencia Artificial y herramientas disruptivas y ciberataques y fraudes informáticos. LawCEDIC responde a esta necesidad ofreciendo una solución integral que permite a los usuarios buscar y analizar documentos de manera estructurada y eficiente, integrando un modelo de inteligencia artificial que optimiza la recuperación de información mediante búsquedas en lenguaje natural.

Este informe documenta todo el proceso de desarrollo de la plataforma LawCEDIC, abarcando no solo su implementación, sino también las decisiones, estrategias y

metodologías utilizadas a lo largo del camino. En él se detalla cómo esta herramienta busca transformar la manera en que se organiza y accede a la información legal, buscando posicionarse como un referente en innovación tecnológica aplicada al ámbito jurídico.

3. Análisis del problema

3.1. Dominio

OCEDIC cuenta con un prototipo de plataforma web destinada a la gestión de documentos judiciales relacionados con cibercrimen y evidencia digital desarrollado en Wordpress, una herramienta ampliamente utilizada para crear y administrar sitios web. Este sistema tiene como objetivo principal optimizar el acceso y consulta de información por parte de abogados y miembros del Poder Judicial. Sin embargo, la plataforma enfrenta importantes limitaciones técnicas y operativas que comprometen su funcionalidad y crecimiento.

Uno de los problemas más significativos es su baja escalabilidad, derivada del uso de tecnologías y técnicas que no permiten una expansión eficiente del sistema. Además, la plataforma presenta una dependencia considerable de herramientas de terceros, que son los *plugins* de Wordpress, lo que afecta su autonomía y sostenibilidad.

En cuanto a la experiencia del usuario, el proceso de búsqueda de documentos resulta poco intuitivo, ya que no cuenta con la capacidad de realizar búsquedas específicas mediante lenguaje natural. En su lugar, se utilizan filtros por temática o tipo de documento, que no satisfacen plenamente las necesidades de los usuarios. A esto se suma que la carga de documentos a la plataforma es completamente manual, un proceso que implica completar múltiples campos, generar resúmenes de extensos documentos judiciales e identificar etiquetas relacionadas con las características de cada archivo resultando así en un procedimiento largo y tedioso en cada carga.

Además, la base de datos utilizada por el sistema actual no es óptima para integrar nuevos servicios, debido a su diseño rígido y limitado. Esto dificulta la implementación de procesos automatizados y de búsqueda avanzada, así como la capacidad de adaptarse a futuras necesidades. Su falta de flexibilidad también afecta el rendimiento y la optimización de consultas, limitando el potencial del sistema para escalar y manejar mayores volúmenes de información y usuarios.

En respuesta a estas limitaciones, el desarrollo de la nueva plataforma LawCEDIC se vuelve crucial para mejorar la carga, gestión y consulta de los documentos judiciales del observatorio.

3.2. Innovación

El aspecto más destacado de la nueva plataforma es la integración de inteligencia artificial para automatizar tareas repetitivas como la carga de documentos con su posterior resumen y extracción de etiquetas, y para mejorar significativamente la experiencia de búsqueda por parte de los usuarios finales. Esta innovación permite abordar problemas previamente mencionados como la dependencia de procesos manuales y la falta de herramientas de búsqueda avanzadas. A través de un modelo de *Retrieval Augmented Generation* (RAG), los usuarios pueden realizar consultas mediante lenguaje natural sobre el repositorio de documentos cargados. Este enfoque, basado en la semántica, permite que los resultados se ajusten al significado e intención de las consultas, simplificando y agilizando el acceso a la información jurídica relacionada con cibercrimen e investigación criminal.

Gracias a estas funcionalidades, la plataforma aspira a convertirse en el primer gran repositorio nacional en implementar inteligencia artificial para la gestión de legislaciones, jurisprudencia y doctrina sobre cibercrimen.

3.3. Requerimientos funcionales

Los requerimientos funcionales mostrados a continuación están agrupados por categoría y ordenados internamente según su relevancia.

3.3.1. Gestión de usuarios

RF01: El sistema debe permitir que se registren nuevos usuarios ingresando un nombre de usuario, un email y una contraseña.

RF02: El sistema debe permitir al usuario acceder a la plataforma web mediante el ingreso de nombre de usuario/email y contraseña.

RF03: El sistema debe permitir que un usuario administrador asigne roles a los usuarios, siendo estos: admin, supervisor, author y user.

RF04: El sistema debe permitir que los usuarios puedan cambiar su contraseña, validando su cuenta con el email registrado.

RF05: El sistema debe permitir que los usuarios con rol de admin puedan desactivar/activar una cuenta registrada.

3.3.2. Gestión de entidades

RF06: El sistema debe contar con una sección “Gestión de Entidades” en la que los usuarios con los roles de admin y supervisor puedan crear, modificar o eliminar entidades.

RF07: El sistema debe permitir que los usuarios con los roles de admin y supervisor puedan crear, eliminar o modificar instancias de la entidad Tipo de Delito. Al crear un Tipo de Delito se deberá especificar un nombre único y opcionalmente un Tipo de Delito superior.

RF08: El sistema debe permitir que los usuarios con los roles de admin y supervisor puedan crear, eliminar o modificar instancias de la entidad Tribunal. Al crear un Tribunal se deberá especificar un nombre único.

RF09: El sistema debe permitir que los usuarios con los roles de admin y supervisor puedan crear, eliminar o modificar instancias de la entidad País. Al crear un País se deberá especificar un nombre único.

RF10: El sistema debe permitir que los usuarios con los roles de admin y supervisor puedan crear, eliminar o modificar instancias de la entidad Órgano. Al crear un Órgano se deberá especificar un nombre único.

RF11: El sistema debe permitir que los usuarios con los roles de admin y supervisor puedan crear, eliminar o modificar instancias de la entidad Fuero. Al crear un Fuero se deberá especificar un nombre único.

RF12: El sistema debe permitir que los usuarios con los roles de admin y supervisor puedan crear, eliminar o modificar instancias de la entidad Eje Temático. Al crear un Eje Temático se deberá especificar un nombre único.

RF13: El sistema debe permitir que los usuarios con los roles de admin y supervisor puedan crear, eliminar o modificar instancias de la entidad Voz/Etiqueta. Al crear una Voz se deberá especificar un nombre único.

RF14: El sistema debe permitir que los usuarios con los roles de admin y supervisor puedan crear, eliminar o modificar instancias de la entidad Norma. Al crear una Norma se deberá especificar un nombre único.

RF15: El sistema debe permitir que los usuarios con los roles de admin y supervisor puedan crear, eliminar o modificar instancias de la entidad Esquema Normativo. Al crear un Esquema Normativo se deberá especificar un nombre y una Norma ó Esquema Normativo superior.

RF16: El sistema debe mostrar, durante la carga de los Esquemas Normativos, un árbol jerárquico que represente las Normas y Esquemas Normativos en relaciones de padres e hijos, facilitando así la visualización de la Legislación.

RF17: El sistema debe mostrar, durante la carga de los Tipos de Delito, un árbol jerárquico que represente las relaciones de padres e hijos.

3.3.3. Gestión de documentos

RF18: El sistema debe permitir la creación de cuatro tipos de documentos: 1. Doctrina 2. Jurisprudencia 3. Recomendaciones, Protocolos y Guías 4. Artículos de Legislación.

RF19: El sistema debe permitir iniciar la creación de un documento de forma "Manual", cargando un PDF e indicando el tipo de documento. Todos los campos estarán vacíos al iniciar este proceso.

RF20: El sistema debe permitir iniciar la creación de un documento de forma "Procesar con Inteligencia Artificial", cargando un PDF e indicando el tipo de documento. La Inteligencia

Artificial procesará el documento y asignará un título, un país, y un fuero (si corresponde), además de mostrar sugerencias de tipos de delitos. También generará un resumen del documento.

RF21: El sistema debe permitir que los usuarios con los roles de supervisor y author puedan crear un documento de tipo Doctrina cargando un PDF y asignando las siguientes entidades obligatorias: Título, Eje Temático, Tipos de Delitos, País, Etiquetas, *Abstract* y Autor. Además, las siguientes opcionales: Fecha, Fecha de Actualización y Subtítulo.

RF22: El sistema debe permitir que los usuarios con los roles de supervisor y author puedan crear un documento de tipo Jurisprudencia cargando un PDF y asignando las siguientes entidades obligatorias: Título, Eje, Tipos de Delitos, País, Etiquetas, *Abstract*, Tribunal y Fuero. Además, las siguientes opcionales: Fecha, Fecha de Actualización y Subtítulo.

RF23: El sistema debe permitir que los usuarios con los roles de supervisor y author puedan crear un documento de tipo Recomendación, Protocolos y Guías cargando un PDF y asignando las siguientes entidades obligatorias: Título, Eje, Tipos de Delitos, País, Etiquetas, *Abstract*, Órgano y Autor. Además, las siguientes opcionales: Fecha, Fecha de Actualización y Subtítulo.

RF24: El sistema debe permitir que los usuarios con los roles de supervisor y author puedan crear un documento de tipo Artículo asignando las siguientes entidades obligatorias: Título, Árbol de Esquema Normativo y *Abstract*. Además, las siguientes opcionales: Subtítulo.

RF25: El sistema debe permitir que los usuarios modifiquen durante la creación de un documento los campos precargados por la Inteligencia Artificial.

RF26: El sistema debe contar con una sección “Gestión de Documentos” en la que los usuarios con los roles de admin y supervisor puedan ver el estado, aprobar, modificar y eliminar los documentos cargados.

RF27: El sistema debe permitir que los usuarios con rol de supervisor puedan editar y eliminar cualquier documento cargado que se encuentre en estado "Pendiente".

RF28: El sistema debe crear automáticamente un borrador del documento en el momento en que se inicie el proceso de carga del archivo.

RF29: El sistema debe permitir guardar los cambios en el borrador durante el proceso de carga del archivo.

RF30: El sistema debe asignar el estado "Pendiente" a los documentos recién creados.

RF31: El sistema debe permitir que los usuarios con los roles de admin y supervisor aprueben un documento con estado "Pendiente", cambiando su estado a "Aprobado", o bien recomiende su edición dejándolo en estado "Pendiente".

RF32: El sistema debe contar con una sección “Mis Documentos” en la que los usuarios con los roles de supervisor y author puedan ver el estado de los documentos de su autoría y, si aún no fueron aprobados, modificarlos o eliminarlos.

RF33: El sistema debe contar con una sección “Mis Borradores” en la que los usuarios con los roles de supervisor y author puedan ver, modificar o eliminar documentos en estado borrador de su autoría.

RF34: El sistema debe permitir que los usuarios con rol de author puedan editar y eliminar los documentos de su autoría que se encuentren en estado "Pendiente".

3.3.4. Búsqueda y visualización de documentos

RF35: El sistema debe permitir que cualquier usuario busque documentos utilizando el modelo de inteligencia artificial. Para ello, en una sección con el nombre “Búsqueda IA” debe ingresar obligatoriamente un texto de búsqueda y, de manera opcional, podrá aplicar filtros de Tipo de Documento, País, Eje, Etiquetas, Tipos de Delitos y/o Autor.

RF36: El sistema debe permitir que cualquier usuario visualice individualmente un documento seleccionado, mostrando sus atributos.

RF37: El sistema debe permitir que cualquier usuario descargue el archivo PDF asociado al documento visualizado.

RF38: El sistema debe mostrar los resultados de los documentos encontrados por la inteligencia artificial y ordenarlos por relevancia de búsqueda.

RF39: El sistema debe contar con una sección "Buscador de Legislación" que permita visualizar el árbol jerárquico creado por las Normas y sus Esquemas Normativos, de manera que los usuarios puedan navegar por él hasta encontrar los artículos en el último nivel del Esquema Normativo seleccionado.

RF40: El sistema debe permitir que cualquier usuario busque documentos sin utilizar IA en una sección “Documentos”, utilizando los filtros de Tipo de Documento, Países, Tipos de Delitos, Ejes y/o Etiquetas, sin hacer uso de la inteligencia artificial.

RF41: El sistema debe contar con una sección "Mis Favoritos" en la que el usuario pueda crear, eliminar o editar directorios y subdirectorios, asignarles un nombre a cada uno, y agrupar dentro de ellos documentos de su interés.

RF42: El sistema debe contar con una sección "Estadísticas" en la que los usuarios con los roles de admin y supervisor puedan ver estadísticas relevantes sobre los documentos cargados, como la cantidad de documentos por atributos.

RF43: El sistema debe mostrar estadísticas de los documentos obtenidos mediante la búsqueda con inteligencia artificial. Ellas son: la cantidad de documentos por Países, la cantidad de documentos por Tipo de Delitos, la cantidad de documentos por Ejes, la cantidad de documentos por Tipo de Documento y el *score* de relevancia de cada documento para la búsqueda actual.

3.4. Requerimientos no funcionales

Los siguientes requerimientos no funcionales están agrupados por categoría y ordenados internamente según su relevancia.

3.4.1. Plazos

RNF01: El plazo de tiempo está acotado por la presentación de la plataforma en un congreso a nivel nacional en San Luis la última semana de noviembre, por lo que debe estar completamente funcional para esa fecha.

3.4.2. Usabilidad y compatibilidad

RNF02: El sistema debe ser desarrollado como una aplicación web.

RNF03: La aplicación web tendrá una interfaz adaptable para los distintos dispositivos, tanto para navegadores como para dispositivos móviles, es decir, debe ser *responsive*.

RNF04: La plataforma debe ser compatible con todos los navegadores principales, como Chrome, Firefox, Safari y Edge, asegurando una experiencia consistente en todos ellos.

RNF05: La plataforma debe contar con la paleta de colores utilizada por OCEDIC en sus publicaciones en redes sociales o en su página web oficial, para mantener la consistencia estética con su identidad institucional.

3.4.3. Mantenimiento y seguridad

RNF06: La aplicación debe contar con un log de errores, *requests* y logueos en el backend, para asegurar el monitoreo continuo y la trazabilidad de las acciones del sistema.

RNF07: El sistema debe garantizar que la información de los usuarios y documentos se almacene de forma segura.

RNF08: La plataforma debe ser mantenible y extensible para la implementación de futuras funcionalidades.

RNF09: Debe utilizarse un sistema de control de versiones para gestionar el desarrollo y mantenimiento del sistema.

RNF10: El sistema debe contar con una función de backup automático de datos, que realice copias de seguridad de la base de datos cada 7 días para evitar la pérdida de información en caso de fallos.

3.4.4. Rendimiento y escalabilidad

RNF11: La plataforma debe garantizar un tiempo de respuesta para las peticiones web que oscile entre 200 milisegundos y 1 segundo en condiciones normales de uso, asegurando una experiencia de usuario fluida y sin interrupciones perceptibles.

RNF12: La plataforma debe garantizar una alta disponibilidad para evitar interrupciones del servicio, asegurando su funcionamiento continuo.

RNF13: La plataforma, en un principio, debe soportar hasta 100 usuarios concurrentes, aunque no se espera alcanzar este nivel en las primeras fases. La arquitectura debe garantizar esta capacidad para asegurar estabilidad y escalabilidad futura.

4. Proyecto

4.1. Objetivo

El propósito principal del proyecto LawCEDIC es desarrollar una plataforma web que permita la gestión eficiente de documentos judiciales vinculados a cibercrimen y evidencia digital. Este sistema busca superar las limitaciones tecnológicas y operativas de la plataforma actual, no solo mediante la incorporación de inteligencia artificial para automatizar la carga de documentos y optimizar la búsqueda de información jurídica, sino también con el uso de tecnologías modernas que garanticen un diseño robusto, escalable y adaptable a futuras necesidades. El objetivo de negocio del referente funcional es comercializar suscripciones que otorguen acceso a la plataforma. Si bien este último no se incorporó al alcance de esta fase del proyecto, se tuvo en cuenta durante el desarrollo para facilitar su implementación futura.

4.2. Alcance inicial

La plataforma LawCEDIC incluirá diversas funcionalidades diseñadas para abordar las necesidades del observatorio y sus usuarios finales. Como funcionalidad principal, se desarrollará un sistema robusto de carga automatizada que permitirá la gestión de documentos judiciales de diferentes características, utilizando la interfaz web. Este proceso estará respaldado por un modelo de inteligencia artificial que identificará etiquetas automáticamente y generará resúmenes de los documentos ingresados.

La plataforma también proporcionará herramientas avanzadas para la consulta y filtrado de documentos. Los usuarios podrán realizar búsquedas específicas en la base de datos sobre ejes temáticos, etiquetas, tipos de documentos o esquemas normativos. Asimismo, se integrará un sistema de búsqueda con inteligencia artificial que permitirá realizar consultas mediante lenguaje natural, mejorando significativamente la experiencia del usuario.

En términos de seguridad, el sistema implementará mecanismos de autenticación para garantizar el acceso controlado y proteger la información almacenada. Por último, se garantizará que la plataforma sea escalable, adaptable a futuras necesidades y accesible

desde diferentes dispositivos, asegurando una experiencia óptima y responsiva para los usuarios.

4.3. Entregables

Al finalizar el desarrollo del proyecto, se entregarán los siguientes:

- Lista de requerimientos funcionales y no funcionales
- Código fuente (backend y frontend) de la aplicación
- Plataforma web funcional
- Documentación:
    - Diagrama de arquitectura del sistema
    - Documentación de la API
- Credenciales de acceso a la máquina virtual junto a los siguientes documentos y archivos:
    - Guía para la administración de los contenedores Docker
    - Scripts de backup y restauración de datos

4.4. Actores

El proyecto contó con la participación de diversos actores, cada uno con un rol particular:

- **Curadores de la información**: está formado por abogados pertenecientes a OCEDIC, interesados en el proyecto de la plataforma web. Son los responsables de cargar a la plataforma fallos recientes vinculados al cibercrimen. Además, realizan la revisión de los datos que devuelve la IA al procesar un determinado documento. Junto a ellos se realizaron diferentes capacitaciones respecto del funcionamiento de la aplicación.
- **Usuarios finales**: la plataforma tiene como público objetivo analistas, abogados, peritos e integrantes del poder judicial. El objetivo es que estos profesionales vean sus tareas simplificadas al poder utilizar las herramientas de búsqueda y gestión de documentos que ofrece LawCEDIC.
- **Equipo de desarrollo**: Su tarea consiste principalmente en incorporar las funcionalidades establecidas previamente y ajustarlas de acuerdo a los comentarios

de las reuniones semanales. Además, el equipo realizó la integración del modelo de inteligencia artificial.

Otros actores de relevancia para el éxito del proyecto fueron:

- **Referente funcional de LawCEDIC**: Javier Vellido, referente funcional del proyecto, fue la cara visible del equipo de curadores especializados de OCEDIC. Era el responsable de transmitir al equipo de desarrollo, semana a semana, *feedback* respecto de las funcionalidades implementadas.
- **Consultora Augmented Experiences**: consultora externa designada por Meta para el proyecto, dedicada a la elaboración de modelos de IA y experiencias inmersivas en realidad virtual. Fueron los encargados de desarrollar el modelo de IA que utiliza LawCEDIC.
- **Directores del trabajo final**: Su rol fue importante en distintas etapas del desarrollo. Su ayuda consistió principalmente en resolver dudas respecto al despliegue de la plataforma y configuración de medidas de seguridad en la máquina virtual.

4.5. Análisis FODA

El análisis FODA es una herramienta utilizada en la gestión de proyectos para evaluar los factores internos (fortalezas y debilidades) y externos (oportunidades y amenazas), que pueden influir en el desarrollo de un proyecto. Este análisis se realizó a priori del desarrollo del proyecto para proporcionar una visión clara de los elementos que favorecen o dificultan el cumplimiento de los objetivos, permitiendo así la toma de decisiones fundamentadas.

Fortalezas:

- Se cuenta con referentes funcionales involucrados en la investigación del derecho y el cibercrimen, sumamente presentes e interesados en el desarrollo del proyecto.
- Experiencia en trabajo conjunto entre los integrantes del equipo.
- Factibilidad de aplicar un esquema de desarrollo ágil para el proyecto que permite una retroalimentación casi instantánea de los *stakeholders* al implementar cada funcionalidad pedida gracias a la disponibilidad e interés de los referentes funcionales y demandantes.
- Conocimientos y experiencia por parte de todos los integrantes del equipo en el

stack MERN (MongoDB, Express, React, Node), idóneo para el proyecto por sus características.

- Buena predisposición por parte del equipo de curadores especializados que cargará los documentos, consciente de la mejora que implica poder automatizar la tarea que ahora están realizando de forma manual.
- Existen algunos repositorios de documentos judiciales genéricos de otros países que pueden ser tomados como referencia.

Oportunidades:

- Inexistencia de un repositorio semejante que comprenda jurisprudencia, doctrina y legislación acerca de cibercrimen a nivel nacional.
- Auge de la inteligencia artificial a nivel mundial que despierta interés en los usuarios finales.
- Relaciones del demandante con la empresa META que permiten facilitar el acceso a LLaMA 3.1, su modelo de inteligencia artificial mejor desarrollado hasta el momento.

Debilidades:

- El equipo de desarrollo tiene conocimientos muy limitados acerca de documentos judiciales y el derecho, por lo que deben invertir tiempo en investigar y realizar una detallada elicitación antes de comenzar el diseño y su posterior implementación.
- Inexperiencia del equipo de desarrollo en un proyecto real de este tamaño.
- Inexperiencia del equipo de desarrollo en inteligencia artificial aplicada al procesamiento de texto, más allá de la formación recibida en la carrera de grado.

Amenazas:

- Demora de la consultora externa al desarrollar el modelo de IA que desencadene un atraso en los plazos de entrega del proyecto.
- Posibles cambios en los requerimientos por parte del demandante por su constante interés en innovar con su producto.
- Aparición de una plataforma similar en el mercado que ofrezca funcionalidades equivalentes o superiores.

Si bien las características mencionadas fueron identificadas al inicio en el análisis FODA, es

posible que, durante o al finalizar el proyecto, surjan nuevas. Esto puede deberse a una falta de experiencia en la gestión de este tipo de proyectos o a la aparición de oportunidades y amenazas que no podían preverse en la fase inicial.

4.6. Análisis de riesgos

Antes de iniciar el desarrollo, el equipo realizó un análisis de riesgos con el objetivo de identificar posibles situaciones críticas que pudieran surgir y definir planes de contingencia específicos para cada una. Este proceso busca reducir la incertidumbre asociada a eventos críticos y mejorar la capacidad de planificación, permitiendo una ejecución del proyecto más estructurada y con menor exposición a imprevistos.

La metodología empleada se basó en la identificación de riesgos potenciales, junto con el análisis de sus posibles consecuencias. Para cada situación identificada, se determinó su probabilidad de ocurrencia y su impacto. Con base en estos dos factores, se evaluó la necesidad de evitar el riesgo y se establecieron planes de contingencia para aquellos casos considerados de alto riesgo.

Para convertir estos conceptos en valores cuantificables, tanto la probabilidad de ocurrencia (P) como el impacto (I) de cada riesgo se clasificaron en una escala de 1 a 3 (donde un valor más alto indica mayor probabilidad o impacto). El peso de cada riesgo se calculó multiplicando estos dos valores. Finalmente, se elaboraron planes de contingencia para los riesgos con un peso igual o superior a 6, ya que representan situaciones con alta probabilidad de ocurrencia y un impacto significativo.

Riesgo	Consecuencia	P	I	Peso
R1: Problemas en la migración de la base de datos existente.	Corrupción de datos. Retrabajo y demoras en el proyecto.	2	3	6
R2: Demora de la consultora externa al desarrollar el modelo de IA.	Atraso en los plazos de entrega del proyecto.	2	3	6

R3: Cambios en los requerimientos del proyecto.	Atraso en los plazos establecidos para el desarrollo.	2	3	6
R4: Obsolescencia del modelo de IA integrado.	Pérdida de utilidad y competitividad frente a futuras plataformas más modernas.	1	3	3
R5: Desinterés del demandante en el proyecto	Demoras en el proyecto.	1	3	3
R6: Obsolescencia de las tecnologías utilizadas para el desarrollo.	Dificultades en la mantenibilidad del proyecto e incorporación de nuevas funcionalidades.	2	2	4
R7: Aparición de una plataforma similar en el mercado que ofrezca funcionalidades equivalentes o superiores	Migración de usuarios hacia la nueva plataforma si perciben que ofrece mayor valor o innovación.	2	3	6

Tabla 1: Análisis de riesgos y consecuencias

Se establecieron entonces planes de contingencia para los riesgos con un peso asociado mayor o igual a 6:

- **R1**: Para simplificar el proceso de migración de datos, se ingadará sobre la estructura de los documentos ya almacenados, cantidad y base de datos utilizada. Esto permitirá una migración eficiente y ordenada.
- **R2**: Al iniciar el proyecto, la consultora proporcionó una planificación, detallando semana a semana los avances y entregables. Se llevarán a cabo reuniones frecuentes con el equipo de Augmented Experiences para analizar el progreso y posibles modificaciones en el modelo.

- **R3**: Se entregará un documento con los requerimientos funcionales y no funcionales al iniciar el proyecto, detallando qué se incluye en el desarrollo de la plataforma.
- **R7**: Se priorizará el cumplimiento estricto de los plazos establecidos, asegurando que la plataforma sea lanzada lo antes posible para capturar una base inicial de usuarios. Además, se diseñará con una arquitectura modular que permita la rápida incorporación de nuevas funcionalidades en función del *feedback* de los usuarios, manteniendo la competitividad en el tiempo.

4.7. Análisis de la competencia

Para comprender el panorama actual en el que se desarrollaría el proyecto, se realizó un análisis de las plataformas existentes en el mercado, guiado por la experiencia del demandante funcional. El proceso comenzó con la identificación de plataformas que ofrecieran características destacadas relacionadas con la gestión de documentos judiciales, tales como búsquedas por filtros o visualización jerárquica de esquemas normativos. Esta selección se realizó en base a las observaciones y recomendaciones proporcionadas por los miembros de OCEDIC, quienes señalaron funcionalidades clave que podrían ser de gran utilidad en su ámbito de trabajo.

Una de las principales páginas web utilizadas en el ámbito judicial a nivel internacional es la de Cornell Law School[1], que permite la visualización jerárquica de documentos de legislación del Código de los Estados Unidos, pero la única manera de buscar a través de él es navegando en la jerarquía o buscando exactamente el título del documento o sección. Además, contiene legislación de temáticas diversas y no está especializada en cibercrimen como se pretende de LawCEDIC.

Otra de las competidoras principales en materia de búsqueda de documentos judiciales es SHERLOC (*Sharing Electronic Resources and Laws on Crime*)[2], que no solo cuenta con legislación y jurisprudencia de múltiples países, sino que los clasifica por temas de interés, entre ellos, ciberdelitos. Sin embargo, al filtrar por estas temáticas y otros filtros manuales como tipo de delito, tribunales, entre otros, los usuarios finales pueden encontrar una

[1] https://www.law.cornell.edu/uscode/text
[2] https://sherloc.unodc.org/cld/en/st/home.html

escasa cantidad de documentos que se ajusten a sus criterios de búsqueda al intentar abarcar distintos ámbitos del derecho.

Tras el análisis, se determinó que no existe una competencia directa que combine la gestión de documentos especializada en cibercrimen, enfocada en la región de Latinoamérica y con una funcionalidad de búsqueda amigable para el usuario a través de lenguaje natural. Esto representa una ventaja competitiva significativa para el desarrollo de la plataforma, ya que responde a una necesidad no cubierta en el mercado actual. Este análisis refuerza la factibilidad y relevancia del proyecto en el contexto actual.

4.8. Planificación

Antes de iniciar el desarrollo del proyecto, se diseñó una planificación detallada con el objetivo de garantizar el cumplimiento de los plazos establecidos y asegurar la calidad final de la plataforma. Este plan incluyó la distribución organizada de tareas y horas entre los tres integrantes del equipo, estimando un promedio de 13 horas semanales por estudiante, lo que resultaría en un total de 1,133 horas de trabajo. Las fases previstas abarcan el período comprendido entre junio de 2024 y enero de 2025. La planificación inicial establecida fue la siguiente:

- **Análisis de requerimientos** (Semanas 1-4, duración total: 90 h): Se llevarán a cabo reuniones con los integrantes del equipo de OCEDIC para relevar las necesidades de la plataforma actual y efectuar un análisis de viabilidad técnica y funcional. Además, se coordinarán encuentros con la consultora de META para interiorizarse en los servicios de IA que se integrarán en el proyecto.
- **Investigación y capacitación** (Semanas 3-7 y 14-16, duración total: 90 h): En esta etapa, se llevará a cabo el estudio de la plataforma previa para comprender su funcionamiento. Asimismo, el equipo profundizará en el *stack* tecnológico seleccionado para el desarrollo y *testing*, y analizará alternativas de despliegue y hosting.
- **Diseño del sistema** (Semanas 5-8, duración total: 60 h): Se definirá la arquitectura del sistema y el *stack* tecnológico a utilizar. También se diseñarán las interfaces de usuario y los principales componentes de la base de datos no relacional (NoSQL).

- **Desarrollo de la plataforma** (Semanas 6-18, duración total: 250 h): Se desarrollará el frontend y el backend de la plataforma utilizando las tecnologías previamente definidas.
- ***Testing*** (Semanas 9-12 y 19-21, duración total: 100 h): Se ejecutarán pruebas manuales y se desarrollarán pruebas automáticas para garantizar la funcionalidad y estabilidad del sistema.
- **Despliegue** (Semanas 15-16, duración total: 50 h): Se configurará el entorno del servidor, incluyendo la base de datos y el hosting, para realizar el despliegue de la plataforma.
- **Migración de la base de datos** (Semanas 16-19, duración total: 91 h): Se llevará a cabo la migración de documentos y metadatos de la plataforma actual hacia el nuevo sistema, asegurando la preservación de los datos.
- **Capacitación del personal** (Semanas 15 y 20, duración total: 20 h): Se planifica la capacitación del equipo de OCEDIC en el uso y mantenimiento de la nueva plataforma.
- **Integración modelo de IA** (Semanas 15-20, duración total: 102 h): Se implementarán y ajustarán los servicios proporcionados por el modelo de IA para integrarlos en la plataforma.
- **Documentación de la plataforma** (Semanas 20-24, duración total: 40 h): Se redactará documentación técnica detallada sobre el código fuente, las APIs desarrolladas y el modelo de IA integrado.
- **Informe y presentación final** (Semanas 23-29, duración total: 240 h): Se prepararán los documentos requeridos por la cátedra y un informe final que resumirá los objetivos alcanzados, las tecnologías utilizadas y el impacto del proyecto.

La planificación de estas fases se estructuró en un cronograma detallado, visible en el siguiente diagrama:

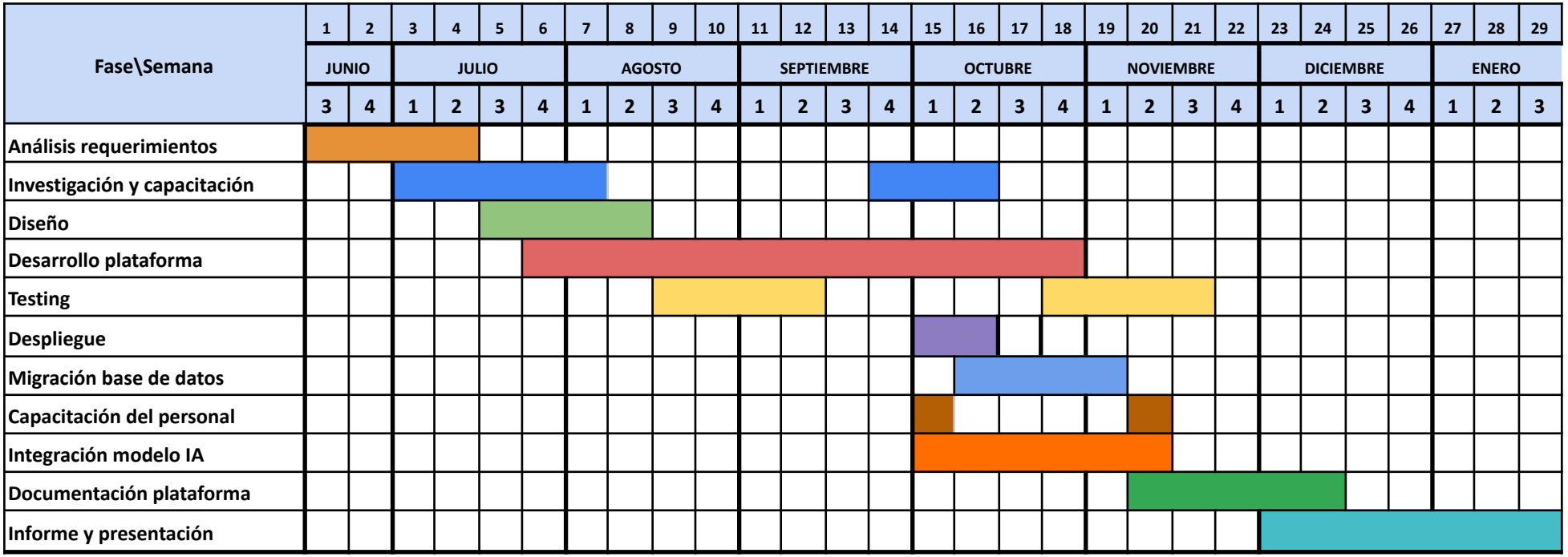

Figura 1: Diagrama de Gantt con la planificación de las tareas del proyecto a lo largo de las semanas

4.9. Metodologías

La metodología empleada en el desarrollo del proyecto fue seleccionada para garantizar una organización eficiente y un enfoque iterativo que permitiera incorporar retroalimentación continua de los diferentes actores involucrados. Se adoptó un enfoque compatible con SCRUM, ajustado a las necesidades específicas del proyecto, utilizando herramientas como Trello para gestionar el *backlog* y realizar el seguimiento del progreso semanal.

SCRUM es un marco de trabajo ágil que se utiliza para la gestión de proyectos complejos, especialmente en desarrollo de software. Está basado en ciclos iterativos y entregas incrementales, denominados "*sprints*", que permiten al equipo adaptarse rápidamente a los cambios y mejorar continuamente. En este caso, aunque el proyecto no utilizó *sprints* formales, se implementó una planificación semanal que cumplía con el principio de iteración y retroalimentación constante.

Además, el uso de GitHub como herramienta de control de versiones se convirtió en un elemento central de la metodología. GitHub permitió gestionar el código fuente del proyecto de manera colaborativa y organizada, asegurando un flujo de trabajo estructurado. En la rama de desarrollo, el equipo integraba las funcionalidades en progreso, mientras que en la rama de producción solo fueron fusionándose los cambios previamente testeados y aprobados por el equipo de desarrollo. Adicionalmente, antes de integrar cualquier funcionalidad, se creaban *pull requests* para revisión, lo que facilitaba la detección de conflictos o errores. Por último, el uso de descripciones claras y estructuradas en los *commits* permitió un seguimiento detallado de los cambios realizados durante todo el desarrollo.

Un componente adicional en la metodología fue el uso de Google Sheets para mantener una bitácora detallada del desarrollo. Este documento sirvió como un registro donde se anotaron las tareas realizadas, el tiempo dedicado a cada una y las fechas correspondientes. El objetivo principal de esta práctica era obtener un registro preciso del tiempo invertido en el proyecto y asegurar la trazabilidad de las actividades. Sin embargo, a pesar de que se le dio relevancia durante el proyecto, no todas las tareas fueron registradas por una falta de

costumbre e inexperiencia por parte de los desarrolladores. Esto representa un error significativo en la gestión del proyecto por lo que es fundamental su reconocimiento y representa un aprendizaje para futuros desarrollos.

4.9.1. Metodología de trabajo

El equipo de desarrollo organizó su trabajo en ciclos iterativos semanales. Cada martes al mediodía, se celebraba una reunión con el referente funcional, Javier Vellido, quien actuaba como enlace entre los analistas especializados de OCEDIC y el equipo técnico. Durante estas reuniones, se presentaban los avances logrados durante la semana anterior, se discutía el estado actual del proyecto y se establecían nuevas funcionalidades o ajustes que debían ser incorporados.

Para gestionar el trabajo, se utilizó un *backlog*, una lista dinámica de todas las tareas y funcionalidades que deben implementarse, la cual se va modificando y actualizando a medida que se avanza en el desarrollo. Este *backlog* fue gestionado a través de Trello, una herramienta de colaboración visual que permite organizar tareas en tableros divididos en columnas. Para la organización de las actividades, se dividió el tablero en tres columnas:

- *Pendientes - backlog*: Tareas planificadas que aún no han sido iniciadas.
- *En curso:* Actividades que se están desarrollando activamente.
- Finalizadas: Funcionalidades terminadas y aceptadas.

En el siguiente gráfico se puede observar la disposición de tareas en Trello a inicios del mes de septiembre:

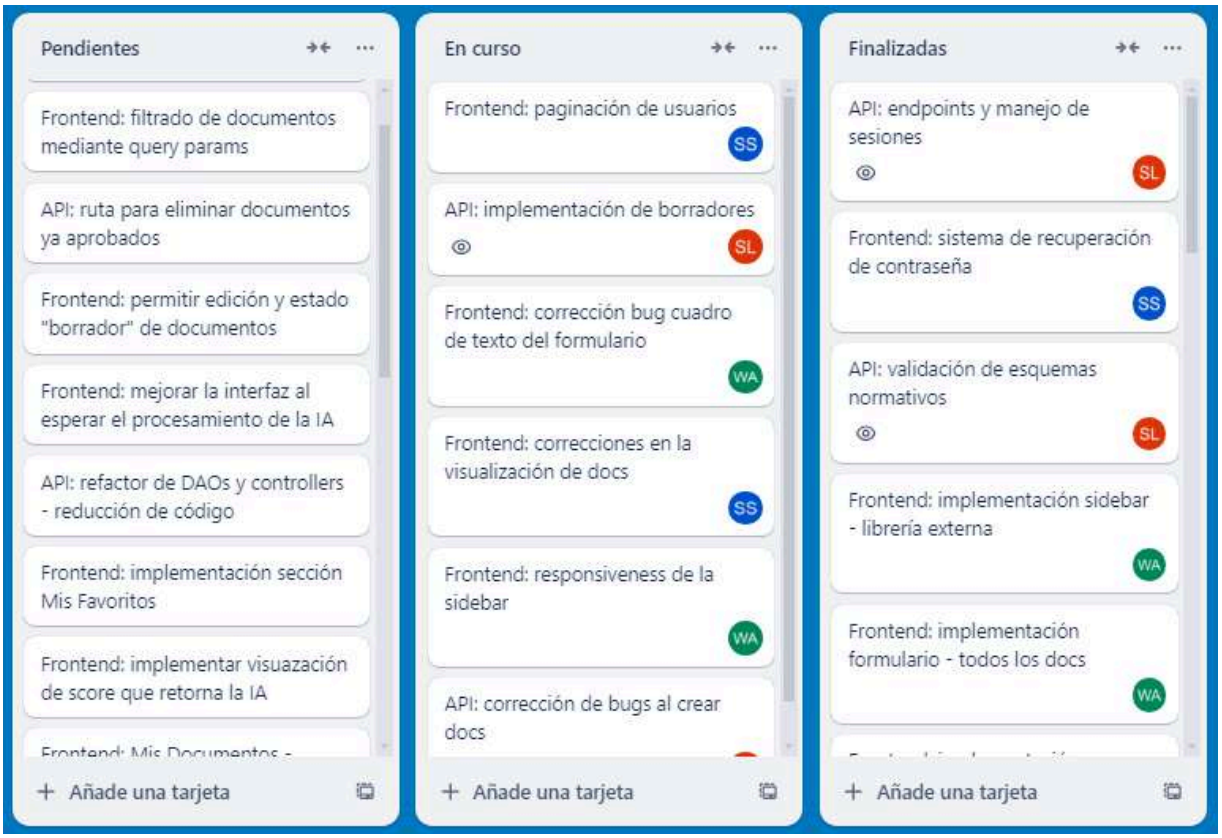

Figura 2: Captura de la disposición de tareas en Trello

4.9.2. Comunicación en el proyecto

La comunicación dentro del equipo y con los distintos actores del proyecto se llevó a cabo mediante diferentes medios:

- **Reuniones semanales con el referente funcional**: Estas reuniones eran el principal canal de comunicación formal entre el equipo de desarrollo y los actores clave del proyecto. Durante las sesiones, el referente funcional proporcionaba observaciones sobre el avance del desarrollo, transmitía las inquietudes y necesidades del equipo de curadores de OCEDIC y determinaba las funcionalidades que debían implementarse.
- **Canales digitales**: Además de las reuniones, se utilizaron canales de comunicación como WhatsApp para resolver dudas rápidas y compartir actualizaciones menores fuera del horario de las reuniones. El equipo de desarrollo realizaba reuniones

frecuentes en Discord para repartir tareas, discutir avances y resolver dudas comunes. Discord es una plataforma de comunicación digital que permite interactuar en tiempo real mediante canales de texto, voz y video. A través de esta plataforma se llevaron a cabo reuniones más informales, las cuales complementaban las reuniones semanales más estructuradas.

- **Tablero Trello**: Sirvió no sólo para gestionar tareas, sino también como un canal adicional de comunicación asincrónica, donde se compartían comentarios y se adjuntaban archivos relevantes.
- **Interacción con la consultora externa de Meta**: La comunicación con Augmented Experiences fue fundamental para garantizar la integración del modelo de IA. Se logró una colaboración eficiente gracias a reuniones técnicas puntuales y consultas específicas realizadas durante el proceso.

A pesar de que había pocas interacciones directas con los abogados de OCEDIC, el rol del referente funcional como intermediario resultó crucial para mantener una comunicación clara y efectiva. Este enfoque evitó ambigüedades y aseguró que las necesidades del equipo de curadores de OCEDIC fueran correctamente interpretadas y traducidas en funcionalidades por parte del equipo de desarrollo.

4.10. Capacitación del equipo de curadores

El proceso de capacitación del equipo de curadores de OCEDIC se llevó a cabo de manera virtual. Se organizaron dos reuniones vía Google Meet en dos etapas distintas del desarrollo de la plataforma, para garantizar una comprensión completa de la misma.

En la primera sesión, se ofreció una introducción general que abarcó las funcionalidades principales, la navegación por la interfaz, la gestión de entidades y la asignación de roles. Durante esta etapa, se brindó acceso inicial a la plataforma, permitiendo a los participantes familiarizarse con sus herramientas y funcionalidades según los roles asignados (autores y supervisores). Esta primera aproximación estuvo orientada a que los usuarios adquirieran una base sólida sobre el uso general de la herramienta.

La segunda sesión se centró en aspectos más avanzados y específicos. En esta etapa, se abordaron los procedimientos para la carga y gestión de documentos, detallando los pasos que los autores deben seguir para incorporar contenido en la plataforma. Además, se explicó el rol de los supervisores, encargados de validar los documentos, así como la importancia de los administradores en la gestión de entidades. Una parte fundamental de esta capacitación fue la introducción al funcionamiento del modelo de inteligencia artificial integrado en la plataforma. Se explicó cómo la IA procesa la información ingresada, cómo genera los resultados relevantes para los usuarios y la importancia de validar estos resultados. El equipo recibió recomendaciones específicas para interpretar las respuestas generadas por el modelo y garantizar la calidad y precisión de la información que se utiliza en las decisiones o procesos posteriores.

Este enfoque estructurado permitió cubrir tanto las necesidades generales como los aspectos técnicos específicos, asegurando que cada miembro del equipo pudiera desempeñar su rol con un conocimiento profundo de las herramientas a su disposición.

5. Diseño del sistema

5.1. Arquitectura del sistema

El diseño del sistema se basa en una arquitectura cliente-servidor que permite una interacción fluida entre los distintos módulos que lo componen. En la *Figura 3*, se presenta un esquema general que ilustra la estructura de la aplicación y la relación entre sus principales componentes. Esta arquitectura fue elegida por su equilibrio entre simplicidad, escalabilidad y facilidad de mantenimiento, asegurando una correcta comunicación entre el frontend, el backend, la base de datos y los servicios externos integrados.

El sistema está compuesto por un frontend desarrollado como una *Single Page Application* (SPA) en React y un backend construido con Node.js y Express. La base de datos utilizada es MongoDB, seleccionada por su flexibilidad en el manejo de datos no estructurados. La comunicación entre el frontend y el backend sigue un esquema RESTful, mientras que la autenticación y autorización se gestionan con JSON Web Tokens (JWT) para garantizar la seguridad de los accesos.

El sistema está desplegado en un servidor VPS alojado en el centro de datos de Ecolan, una cooperativa local de Batán. La implementación con Docker y Docker Compose permite una gestión eficiente de los contenedores y una mayor escalabilidad. Además, la plataforma se conecta con un sistema de Inteligencia Artificial, desplegado en AWS Bedrock, que hace uso de un modelo de IA basado en Retrieval-Augmented Generation con LLaMA 3.1. Esta arquitectura permite un rendimiento eficiente, una gestión segura de la información y una base sólida para futuras mejoras y expansiones.

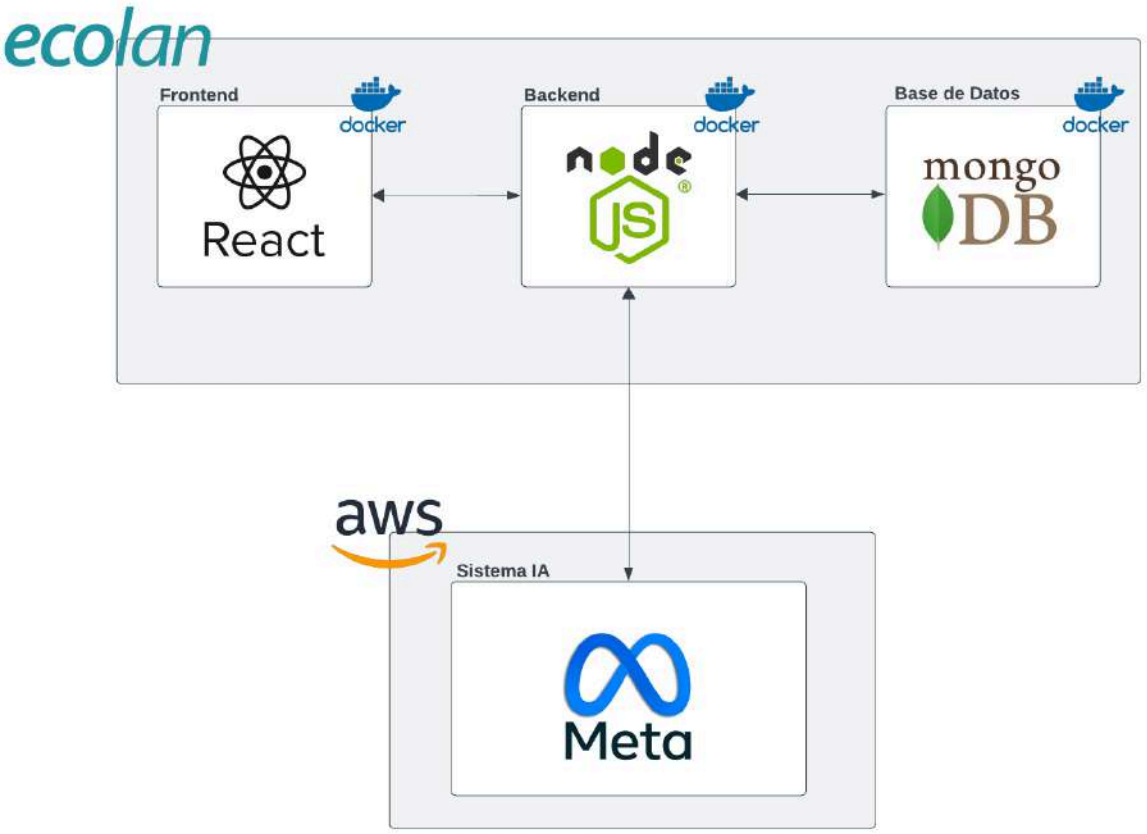

Figura 3: Arquitectura del sistema

5.2. Backend

5.2.1. Tecnologías

Para la elección del *stack* tecnológico se consideró esencialmente la experiencia de los desarrolladores en tecnologías web y su compatibilidad con la solución de despliegue convenida. Por lo tanto, se determinó la utilización del *stack* MERN, compuesto por las tecnologías MongoDB para la base de datos, Express para la API del backend, React para el frontend y Node.js como entorno general de ejecución. Además de la previa experiencia, otra ventaja de su uso es el exponencial crecimiento en los últimos años, que consolidó una activa comunidad de desarrolladores, quienes han enfrentado desafíos similares o extrapolables.

La versión de Node.js utilizada fue la LTS (*Long Term Support*) 18.12.0 y el lenguaje de programación para la API fue JavaScript, muy común en desarrollos web. Además, para la

implementación del proyecto se utilizaron múltiples librerías y dependencias que no existen nativamente en el entorno de Node, pero que sí son proporcionadas por terceros y permiten resolver de forma rápida y sencilla varios de los módulos planteados.

5.2.2. Dependencias fundamentales

A continuación se detalla el listado de dependencias de terceros utilizadas para la implementación con un breve resumen de la razón de su utilización. Cabe resaltar que para su instalación se usó el sistema de gestión de paquetes por defecto de Node.js, npm (*Node Package Manager*).

Dependencias relacionadas a los protocolos de comunicación:

- **Express.js**: Framework minimalista para crear servidores web y manejar rutas de forma sencilla. Es la base para configurar el backend de la aplicación.
- **Axios**: Cliente HTTP que permite realizar solicitudes a servicios externos, manejando peticiones y respuestas. Se utiliza para la comunicación con el servicio de inteligencia artificial.
- **CORS**: *Middleware* que habilita el intercambio de recursos entre diferentes orígenes (*Cross-Origin Resource Sharing*), necesario para permitir la comunicación entre el frontend y backend en distintos dominios.

Dependencias relacionadas a la seguridad:

- **bccrypt**: Librería para encriptar contraseñas de manera segura y comparar *hashes*. Se utiliza para garantizar la confidencialidad de las credenciales de los usuarios.
- **Dotenv**: Maneja las variables de entorno de forma segura. Permite almacenar información sensible (como claves secretas y credenciales) fuera del código fuente.

Dependencias relacionadas al manejo de sesiones:

- **JSON Web Token**: Librería para crear y verificar tokens JWT (*JSON Web Tokens*), usados para la autenticación y autorización en la aplicación.
- **Passport**: Middleware de autenticación que soporta múltiples estrategias (como autenticación local u OAuth). Facilita la gestión de inicios de sesión.

- **cookie-parser**: Middleware para analizar cookies en las solicitudes HTTP. Es necesario para manejar la autenticación basada en cookies o sesiones.

Dependencias relacionadas al *testing*:

- **chai**: Librería de aserciones para escribir pruebas de forma clara y legible. Permite validar que las funciones y módulos funcionan correctamente.
- **mocha**: Marco de pruebas para JavaScript que organiza y ejecuta casos de prueba en el backend.
- **supertest**: Herramienta para realizar pruebas de integración en servidores HTTP.

Otras dependencias:

- **form-data**: Facilita la estructuración y envío de los archivos PDF hacia la API que contiene el modelo de inteligencia artificial.
- **mongoose** y **mongoose-paginate-v2**: Conecta el entorno de ejecución de Node.js con el de MongoDB proporcionando una solución basada en esquemas aunque la base de datos no sea relacional.
- **multer**: Middleware para el simple manejo y recepción de archivos (en este caso, documentos pdf).
- **nodemailer**: Facilita el envío de emails. Utilizado para enviar el email de recuperación de contraseña a usuarios que la hayan olvidado o cuya cuenta se haya bloqueado.
- **swagger-jsdoc** y **swagger-ui-express**: Crea un *endpoint* en el cual se muestra una página web con toda la documentación de la API de forma amigable y permite al usuario probar sus distintos *endpoints* con unos pocos clicks.
- **validator**: Validación y sanitización de cadenas de caracteres. Utilizado para el registro de usuarios y validación de sus emails y contraseñas.
- **winston**: Librería de *logging* para estandarizar los informes de errores y *logs* generados.

Con el objetivo de garantizar la confiabilidad y escalabilidad del sistema, se seleccionaron dependencias con comunidades de desarrollo activas y amplio mantenimiento, asegurando actualizaciones y soporte continuo. La compatibilidad entre las dependencias fue un criterio

clave, evitando conflictos y facilitando la integración. Estas buenas prácticas sientan las bases para un sistema robusto y adaptable, capaz de evolucionar con las necesidades futuras sin comprometer el rendimiento.

5.2.3. Componentes de la API

Para garantizar la calidad de la plataforma es necesario poner el foco en diferentes atributos de calidad. En esta sección se detallarán las decisiones de diseño tomadas en pos de mejorar la mantenibilidad del sistema, su escalabilidad y la fácil detección de errores. De esta manera, el proyecto queda encaminado para un crecimiento controlado en nuevas funcionalidades y en el alcance de usuarios, alineado con los objetivos del negocio del cliente

En la *Figura 4* se puede observar un gráfico con los componentes del sistema, que permite visualizar la división en directorios que contienen los archivos de código de la API.

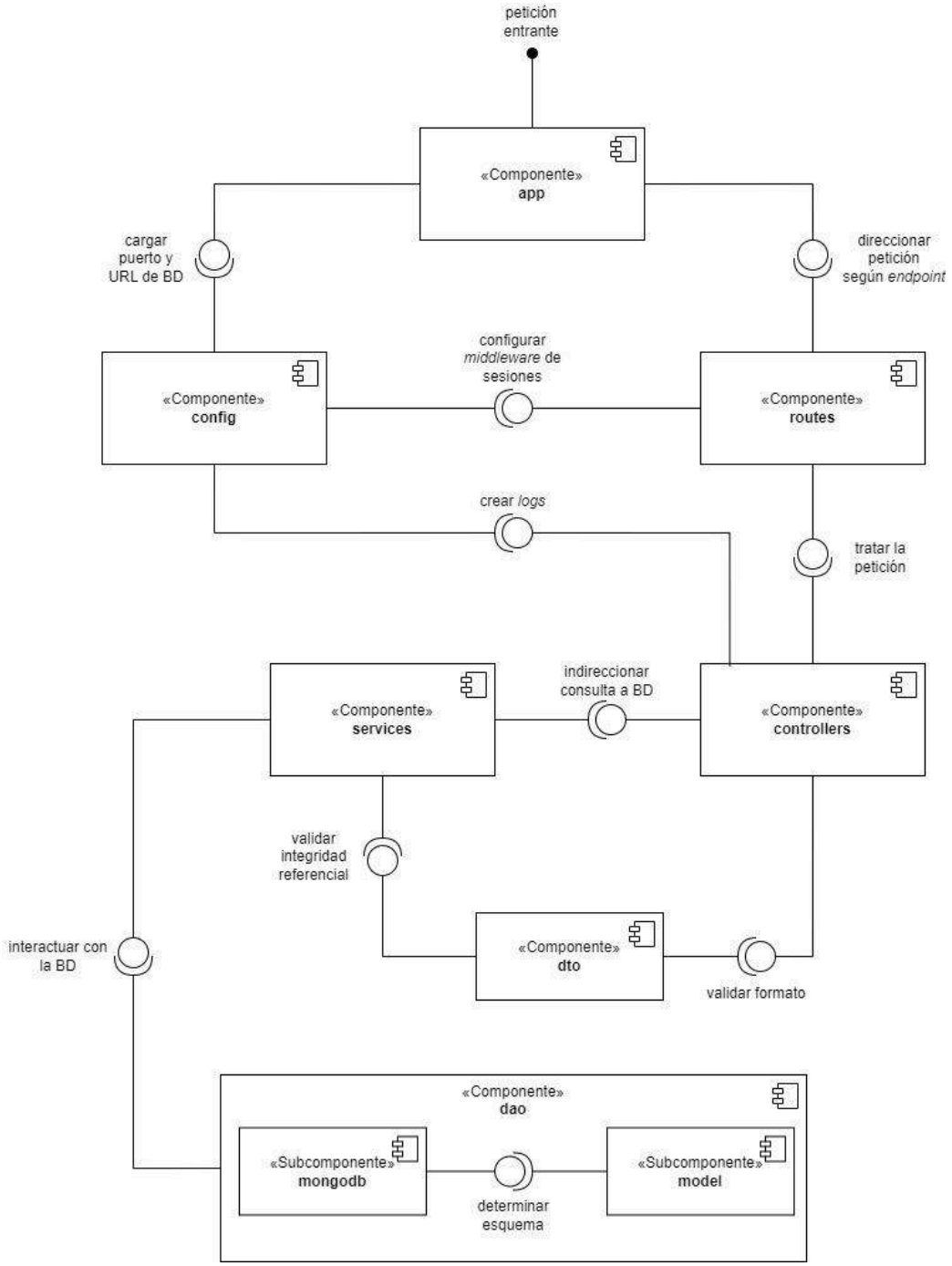

Figura 4: Diagrama de componentes del backend

- **app**: Es el archivo que utiliza la librería Express.js para crear, en pocas líneas de código, un servidor que espera peticiones. Se comunica con los diferentes *routers* para direccionar las peticiones entrantes de acuerdo al *endpoint* en el cual se reciben.

- **config**: Contiene archivos de configuración de varias librerías para luego ser utilizados en distintas partes del código:
    - **dotenv.config**: Utiliza la librería dotenv para cargar las variables de entorno del archivo .env, encapsularlas en un objeto y exportarlo para ser accedido en otros ámbitos.
    - **logger.config**: Configura un *middleware* que agrega a la petición la funcionalidad de generar y almacenar *logs* para luego ser utilizado por los controladores en caso de que ocurra algún error interno del servidor.
    - **passport.config**: Configura un *middleware* con las estrategias de la librería passport utilizadas para registrar nuevos usuarios e iniciar sesión utilizando manejo de *cookies* y *JSON Web Token*.
- **routes**: Contiene los archivos con las rutas a los distintos *endpoints* de la API. Un archivo *router* crea y ordena varios *middlewares* para estandarizar el cuerpo de las respuestas y errores y para verificar que el usuario que envió la petición esté autenticado y tenga los permisos necesarios para acceder al recurso que solicitó. En el Anexo 10.1: Documentación de la API Rest se encuentran los *endpoints* expuestos por el servidor.
- **controllers**: Contiene los archivos encargados de procesar las peticiones recibidas en los distintos endpoints, realizar las validaciones necesarias y delegar las tareas a los servicios correspondientes. Finalmente, responden a los remitentes con el recurso solicitado o un mensaje de error adecuado. Cada ruta cuenta con su controlador correspondiente.
- **services**: Funciona de intermediario entre el controlador de un recurso y el código que establece la comunicación directa con la base de datos. Es una capa adicional que se agrega para facilitar cambios en el tratamiento de los datos antes de enviarlos o luego de recibirlos de la base de datos. De esta forma se permitiría, por ejemplo, cambiar el motor de base de datos o la forma de realizar las consultas para optimizarlas sin impactar directamente en los módulos relacionados con recibir y responder las peticiones.
- **dto**: Se tiene un archivo por estructura almacenable en la base de datos. Son clases encargadas de recibir los campos de los objetos a almacenar, validar su formato e integridad referencial y se instancian antes de crear o actualizar una entrada en la

base de datos. De esta manera se facilita la especificación de los errores de envío de datos por parte del usuario que envió la petición y son independientes del sistema de gestión de base de datos utilizado.

- **dao**: Contiene los archivos de código que funcionan como interfaz directa con la base de datos. Se estructura en dos subdirectorios:
    - **model**: Cada archivo determina el esquema de los documentos a almacenar en la base de datos MongoDB ya que, si bien es no relacional, se establece una estructura validada por el propio motor. Para ello, se utiliza la librería mongoose. Además, se configuran algunos *triggers* y *stored procedures*.
    - **mongodb**: Es la interfaz entre los servicios antes mencionados y la base de datos MongoDB. Contiene la funcionalidad que realiza la consulta directa a la base de datos utilizando los esquemas establecidos en el módulo *model*.

5.2.4. Seguridad

En una plataforma como LawCEDIC en la que uno de los objetivos del negocio es la comercialización de un servicio, la seguridad es fundamental. Lo importante no es simplemente resguardar el acceso a la información presente en la plataforma, sino protegerla de accesos indeseados que puedan ocasionar peticiones no autorizadas al modelo de inteligencia artificial. La razón de esta prioridad es que el principal costo es la *tokenización* del texto en lenguaje natural, por lo que solo deberían tener acceso a esta funcionalidad quienes tengan un usuario autorizado.

5.2.4.1. Autenticación y credenciales

En primer lugar, fue necesario diseñar una autenticación y manejo de credenciales robustas. Para el proceso de registro de usuarios se utilizó la librería mongoose y bcrypt para almacenar las credenciales de acceso en la base de datos. Además, mediante un *middleware*, la contraseña se cifra automáticamente con una clave secreta, almacenando un hash en lugar de la contraseña en texto plano. Luego, en cada inicio de sesión se utiliza esa misma clave para cifrar la contraseña enviada por el usuario y compararla con el *hash* guardado en la base de datos para autenticar exitosamente o rechazar al usuario. Para mantener de forma segura la clave secreta de encriptación se utilizó la librería dotenv, que permite almacenar variables de entorno en forma segura en un archivo llamado “.env”.

Además, para evitar ataques de fuerza bruta se estableció que cada usuario tenga cuatro oportunidades para escribir correctamente sus credenciales de usuario y al quinto intento se bloquea la cuenta temporalmente. Luego de ese tiempo, podrá volver a intentarlo.

5.2.4.2. Mantenimiento de sesiones activas

Registrar usuarios e iniciar sesión no es suficiente si la plataforma no puede mantener activa la sesión del usuario en el navegador o la aplicación cliente durante un tiempo determinado. De lo contrario, el usuario tendría que autenticarse en cada petición, lo que afectaría la experiencia de uso. Esto ocurre debido a la naturaleza *stateless* del protocolo HTTP utilizado en la comunicación entre cliente y servidor.

Para evitar que los usuarios deban ingresar su nombre y contraseña en cada solicitud, se implementó un mecanismo basado en la librería JSON Web Token (JWT). Cuando un usuario inicia sesión en un cliente web, un middleware envía en la respuesta una *cookie* con un *token* de autenticación, que se almacena junto al nombre de usuario y su rol. De este modo, cada vez que se realice una nueva petición, el *token* se enviará automáticamente, permitiendo al servidor reconocer que el usuario tiene una sesión activa. Esta *cookie* se firma con JWT utilizando una clave secreta y tiene un tiempo de expiración definido (por ejemplo, 1 hora). Posteriormente, el servidor valida que el *token* no haya sido alterado ni haya caducado. Además, para mejorar la usabilidad, si se recibe una petición de un usuario cuyo token está próximo a expirar, JWT permite renovarlo automáticamente, evitando que el usuario deba iniciar sesión nuevamente de manera innecesaria.

5.2.4.3. Roles y permisos

Además de la autenticación también es fundamental la autorización, es decir, qué funcionalidades del sistema puede utilizar cada usuario. Alguien que quiera acceder a un *endpoint* de la plataforma sin haber iniciado sesión recibirá un error 401 (*unauthorized*) mientras que alguien con una sesión activa que quiera acceder a un *endpoint* para el cual no tiene permisos recibirá un error 403 (*forbidden*). Esta discriminación entre roles se realiza mediante la clasificación de los *endpoints* y un *middleware* que analiza los *headers* de la petición para verificar los permisos del usuario.

5.2.5. Testing

Para comprobar el correcto funcionamiento de la API se realizaron 73 *tests* automáticos que se detallarán luego. Sin embargo, en un comienzo se realizaron pruebas manuales que permitían avanzar sin detener demasiado el desarrollo. Estas pruebas manuales se realizaron utilizando la herramienta Postman que ofrece funcionalidades sencillas para probar los diferentes *endpoints*. La codificación de los *tests* automáticos fue incremental, a medida que se iba finalizando la implementación de cada funcionalidad. Primero se escribieron las pruebas de inicio de sesión y de registro de usuarios; luego los de las altas, bajas y actualizaciones de las diferentes entidades almacenadas en la base de datos como países, fueros y etiquetas y finalmente todo lo relacionado a manipulación de documentos judiciales.

Los *tests* mencionados se implementaron con las librerías chai, mocha y supertest. La primera permite escribir aserciones para verificar que los valores de las pruebas sean los esperados y se utilizó principalmente para validar respuestas HTTP y las propiedades de los objetos retornados. El segundo paquete consolida el *framework* que facilita la definición de *suites,* la medición del tiempo de ejecución y la estructuración de distintos *test cases* en lotes de pruebas más pequeñas manejando distintos niveles de granularidad. Por último, supertest es una herramienta para realizar solicitudes HTTP en pruebas que permite simular interacciones con la API de forma sencilla y verificar las respuestas. La elección de esta combinación de funcionalidades permitió la implementación de un flujo de pruebas eficiente y legible para verificar el funcionamiento de la plataforma de forma automática.

El *test* de integración resultante comprende la siguiente secuencia, siempre considerando los permisos para cada rol de usuario:

1. Prueba de documentos por tipo (se realiza para un artículo, una legislación, una doctrina y una guía/protocolo)
    a. Creación del documento
    b. Obtención del documento
    c. Obtención de todos los documentos de ese tipo
    d. Obtención de todos los documentos de ese tipo utilizando filtros y paginación
    e. Actualización del documento previamente creado

2. Prueba de documentos generales
    a. Obtención de un documento pendiente
    b. Aprobación de un documento pendiente
    c. Obtención de todos los documentos pendientes
    d. Obtención de todos los documentos aprobados
    e. Obtención de todos los documentos de todos los tipos
    f. “Desaprobación” de un documento ya aprobado
    g. Obtención de los documentos creados por un author o supervisor con una sesión activa
    h. Eliminación de un documento creado por el usuario *logueado*
3. Prueba de entidades (cada una de las siguientes pruebas se realizó para los países, tribunales, fueros, órganos, ejes temáticos y etiquetas)
    a. Creación de la entidad
    b. Eliminación lógica y alta lógica luego de eliminarla
    c. Obtención de la entidad con diferentes filtros
        i. Obtención de los no eliminados
        ii. Obtención de todos, incluyendo eliminados
        iii. Paginación de los no eliminados
4. Prueba de esquemas normativos
    a. Creación de un tipo de norma (esquema normativo raíz del árbol)
    b. Obtención de todos los tipos de norma
    c. Obtención de un tipo de norma
    d. Eliminación lógica de un tipo de norma
    e. Imposibilidad de crear un esquema normativo hijo de un tipo de norma eliminado
    f. Alta luego de la eliminación lógica de un tipo de norma existente
    g. Creación de un esquema normativo hijo de un tipo de norma
    h. Actualización de un esquema normativo
    i. Obtención de todos los hijos de un esquema normativo
5. Prueba de sesiones
    a. Registro de un nuevo usuario
    b. Inicio de sesión

   c. Obtención de información del usuario con una sesión activa
   d. Cierre de sesión
6. Prueba de usuarios
   a. Obtención de todos los usuarios
   b. Paginación de todos los usuarios
   c. Obtención de un único usuario
   d. Cambio de rol de un usuario
   e. Eliminación lógica de un usuario
   f. Imposibilidad de iniciar sesión con un usuario eliminado

Como puede apreciarse, si bien no se realizó explícitamente TDD (*Test Driven Development*), se abarcaron satisfactoriamente la mayoría de los requerimientos funcionales a la hora de la codificación de las pruebas. No obstante, superar las pruebas no fue un desafío fácil, pero se logró con éxito. La falta de experiencia en la escritura de pruebas provocó que muchas veces estas fallaran porque no estaban correctamente escritas y no por un fallo en el código de la API.

5.3. Frontend

5.3.1. Tecnologías

Para el desarrollo del frontend de la plataforma, se utiliza React en su versión 18, que es una librería JavaScript de código abierto, creada y mantenida por Facebook, ahora META, junto con la comunidad de software libre. Junto a React se utiliza Vite como entorno de desarrollo local, que es una herramienta moderna de empaquetado de archivos y desarrollo rápido que permite una inicialización casi instantánea del proyecto, así como actualizaciones en tiempo real (Hot Module Replacement) durante el desarrollo. Esta combinación ofrece un flujo de trabajo más eficiente y ágil. En cuanto a React, es ampliamente reconocida en la industria por su capacidad para construir interfaces web personalizadas, dinámicas y escalables, gracias a su manejo eficiente de componentes HTML, CSS y JavaScript. Al tratarse de una plataforma web, React permite crear componentes reutilizables y ofrecer una experiencia de usuario fluida y moderna.

Además, React se integra fácilmente con Node.js, la tecnología empleada en el backend, formando parte del stack MERN previamente mencionado. Esta integración garantiza una correcta comunicación entre las diferentes capas del sistema, mejorando la cohesión del proyecto y facilitando el desarrollo.

Por último, la elección de esta librería también responde a la experiencia previa del equipo de desarrollo, que cuenta con un sólido conocimiento en su uso. Esto asegura una implementación ágil y efectiva, maximizando la calidad del producto final.

5.3.2. Dependencias fundamentales

En el desarrollo del frontend de la aplicación, se aprovechó la flexibilidad de React para integrar una variedad de bibliotecas externas que permitieron optimizar el proceso de desarrollo, mejorar la experiencia del usuario y garantizar un diseño funcional y atractivo.

Cada una de estas dependencias cumple un propósito específico, ya sea en la funcionalidad del producto, el diseño visual o en herramientas de desarrollo que facilitan la colaboración y la calidad del código. A continuación, se describe brevemente el uso de cada una de estas bibliotecas.

Dependencias de desarrollo:

- **vite**: Herramienta de construcción rápida y ligera para aplicaciones web modernas. Se utiliza para optimizar el desarrollo y compilación del frontend.
- **tailwindcss**: Framework de CSS basado en utilidades que permite un diseño rápido y personalizado directamente en las clases de los elementos HTML, optimizando la creación de estilos consistentes y reutilizables.

Dependencias de producción:

- **react-router-dom**: Biblioteca para la gestión de rutas en aplicaciones React. Permite la navegación entre diferentes vistas de la aplicación de manera declarativa y sin recargar la página.
- **axios**: Biblioteca para realizar peticiones HTTP. Se utiliza para interactuar con las APIs del backend, facilitando la obtención y envío de datos desde el frontend.

- **react-pro-sidebar**: Biblioteca utilizada para implementar la barra lateral principal de la aplicación, proporcionando una solución efectiva y personalizable para la navegación.
- **react-select**: Herramienta para la creación de inputs de tipo "select" personalizables y accesibles, mejorando la experiencia del usuario en formularios.
- **react-icons**: Biblioteca que proporciona una amplia variedad de iconos predefinidos, utilizada para los iconos generales de la aplicación.
- **react-dropzone**: Librería que facilita y mejora la experiencia del usuario al cargar archivos, como documentos PDF, mediante una interfaz interactiva de arrastrar y soltar.
- **react-slick**: Utilizada para implementar carruseles o "carousels" que muestran actualizaciones como contenido rotativo e interactivo en la interfaz.
- **react-chart.js-2**: Biblioteca que ofrece gráficos dinámicos y personalizables, como gráficos de barras o gráficos de pastel, utilizados para mostrar las estadísticas mencionadas en los requerimientos en la aplicación.
- **icomoon-react**: Herramienta necesaria para cargar y utilizar los logos en el formato específico brindado por OCEDIC.
- **ts-particles**: Librería visual que permite mostrar animaciones de partículas en la pantalla, mejorando la estética de la interfaz en pantallas de carga.
- **tiptap**: Biblioteca para implementar un editor de texto enriquecido (RTE, Rich Text Editor), que permite a los usuarios de carga formatear texto con opciones avanzadas como listas, enlaces y alineaciones.

En el desarrollo del frontend, la selección de dependencias actualizadas es vital para garantizar una experiencia de usuario óptima y un rendimiento eficiente. React, como ecosistema dinámico, se beneficia de bibliotecas especializadas que facilitan la creación de componentes y funcionalidades complejas. No obstante, es crucial asegurar su compatibilidad para evitar conflictos y mantener la estabilidad del sistema. Por ello, se priorizó la elección de dependencias con comunidades activas y amplio mantenimiento, asegurando así su vigencia y la disponibilidad de soporte continuo.

5.3.3. Estructura del Frontend

El frontend de la plataforma está diseñado utilizando el enfoque de componentes que ofrece React. Aunque React permite trabajar tanto con clases tradicionales como con componentes funcionales, el enfoque basado en componentes es el más moderno y ampliamente adoptado en la actualidad. Este enfoque permite encapsular la lógica y la interfaz de usuario, facilitando el mantenimiento y la expansión del sistema.

La estructura del frontend está organizada en dos carpetas principales que son "componentes" y "páginas". La carpeta de componentes incluye elementos que representan partes específicas de la interfaz, como formularios, tablas o selectores. Por su parte, las páginas agrupan varios componentes para formar vistas completas que se asocian a rutas específicas de la aplicación.

Para la navegación entre páginas, se utiliza React Router, biblioteca previamente mencionada que permite gestionar las rutas de la aplicación. Esto garantiza que cada página esté vinculada a una URL única, facilitando tanto la navegación del usuario como la organización del código.

5.3.4. Seguridad

Aunque la capa principal de seguridad de la plataforma se maneja en el backend, el frontend también implementa medidas específicas para garantizar una experiencia segura para los usuarios y proteger la información.

Uno de los mecanismos clave es el uso de interceptores de *requests*, que permiten detectar respuestas HTTP con errores de autorización, como los códigos 401 (*Unauthorized*) o 403 (*Forbidden*). En estos casos, el sistema redirige automáticamente al usuario a una página que informa que no está autorizado para realizar esa acción. Además, cuando el *token* de autenticación del usuario expira, el sistema lo desloguea automáticamente, evitando accesos no válidos con sesiones caducadas.

La interfaz también se asegura de limitar el acceso a las diferentes funcionalidades según el rol del usuario. Las páginas o secciones a las que un usuario no tiene permiso de acceder no

se muestran en la navegación o el menú principal. Sin embargo, si el usuario intenta acceder directamente a una página restringida, como la de gestión de documentos, será redirigido a una página de acceso denegado. Además, cuando un usuario inicia una request desde la interfaz, el sistema valida sus permisos para determinar si cuenta con el rol adecuado para realizar esa acción, asegurando que las restricciones se mantengan incluso en caso de intentos directos de acceso. Estas medidas garantizan que cada usuario solo pueda interactuar con las secciones del sistema que están autorizadas para su rol.

Por último, el frontend evita procesar o almacenar información crítica directamente. En lugar de ello, delega estas responsabilidades al backend, que se encarga de aplicar las medidas de seguridad necesarias. Esto minimiza el riesgo de exposición de datos sensibles a través de la interfaz de usuario.

5.3.5. Testing

El testeo en el desarrollo del frontend se realiza principalmente de forma manual al implementar nuevas funcionalidades. Esto se debe a que la creación de tests automáticos para toda la interfaz requiere una extensa interacción con el *DOM* y el uso de librerías como Jest, lo cual puede demandar una cantidad significativa de tiempo. Dado que las necesidades del cliente priorizaban tiempos de desarrollo más ajustados, se decidió enfocar los esfuerzos en implementar un robusto conjunto de tests automáticos en el backend, garantizando la correcta funcionalidad de las API que comunican ambos extremos del sistema.

Los tests manuales se llevan a cabo una vez finalizada la incorporación de una nueva funcionalidad. Durante este proceso, se realiza un rastreo exhaustivo para verificar que los cambios no afecten negativamente a otros módulos del sistema con los que la nueva funcionalidad interactúa. Además, se prueban escenarios con inputs incorrectos para comprobar que se muestran los mensajes de error correspondientes y se valida que la información enviada cumpla con el formato esperado. Para estas verificaciones, se utilizó la consola de desarrollo del navegador.

Dado que la plataforma es responsiva, también se realizan pruebas en distintas resoluciones de pantalla para garantizar una experiencia visual adecuada. Estas pruebas abarcan pantallas *Full HD* y *HD*, así como dispositivos móviles, asegurando que la interfaz se adapte correctamente a cada formato.

5.4. Integración con sistema de Inteligencia Artificial

El elemento innovador de la nueva plataforma LawCEDIC es la integración de un sistema de inteligencia artificial avanzado basado en *Retrieval-Augmented Generation* (RAG), combinado con el *Large Language Model* (LLM) LLaMA 3.1, reconocido por su capacidad de comprensión contextual y generación de texto de alta calidad. La arquitectura RAG aprovecha las capacidades de este modelo para optimizar tanto la indexación como la recuperación de documentos. Por eso, el sistema almacena la información, la comprende y la contextualiza, ofreciendo resultados más precisos y relevantes para los usuarios.

El sistema de inteligencia artificial fue desarrollado por la consultora Augmented Experiences, delegada por META. Durante el proceso, se mantuvo una comunicación constante con dicha consultora para lograr una adaptación mutua entre la plataforma web y el modelo provisto. Además, se trabajó en la incorporación de funcionalidades adicionales al modelo de inteligencia artificial inicial, basadas en los requerimientos específicos planteados por OCEDIC.

Para su integración con LawCEDIC, se utiliza una API proporcionada por Augmented Experiences, la cual ofrece tres servicios diferenciados. Esta API incluye una *API-Key* que permite su uso de forma autenticada, garantizando un acceso seguro y controlado a los servicios de inteligencia artificial.

5.4.1. Servicios del sistema de inteligencia artificial

Los servicios proporcionados por el sistema de inteligencia artificial están diseñados para automatizar tareas repetitivas y demandantes de tiempo, facilitando significativamente el trabajo del equipo de OCEDIC encargado de gestionar y cargar documentos en la plataforma.

Este sistema se encarga tanto de almacenar los documentos en formato PDF como de optimizar su búsqueda mediante inteligencia artificial, devolviendo resultados relevantes a las consultas realizadas en el buscador.

5.4.1.1. Servicio de extracción de datos de documentos

El servicio de extracción de datos relevantes a partir de documentos en formato PDF permite a los usuarios cargar nuevos documentos en la plataforma. Estos documentos son almacenados en una base de datos y se proporciona una URL para su descarga dentro de la plataforma. Los datos extraídos por el modelo de IA están alineados con las entidades definidas en los requerimientos del sistema, asegurando precisión y coherencia en la información procesada. Además, los usuarios pueden modificar o agregar atributos a un documento previamente cargado para adaptar la indexación a sus necesidades.

Inicialmente, todos los documentos debían ser procesados por el modelo de IA para extraer y generar sus datos, permitiendo luego realizar búsquedas con ellos. Sin embargo, esta modalidad no resultó eficiente en todos los casos. Para algunos documentos, ya se disponía del resumen y de la mayoría de los datos requeridos, haciendo innecesario el procesamiento automático. Además, en documentos de gran tamaño, el análisis por IA podía ser un proceso demandante y prolongado.

Para solucionar este problema, se incorporó una funcionalidad que permite a los usuarios elegir entre dos opciones al cargar un documento:

1. **Procesamiento automático con IA**, donde el modelo analiza, genera y devuelve los datos relevantes de manera automática.
2. **Carga manual de datos**, donde el usuario introduce directamente los datos necesarios para que el modelo de IA pueda buscar el documento, pero omitiendo el procesamiento inicial del modelo de inteligencia artificial.

Al utilizar este servicio, se crea una entrada del documento en la base de datos del servicio de Inteligencia Artificial con los campos necesarios para su posterior búsqueda. Esta entrada es diferente a la del documento creado en la plataforma LawCEDIC, ya que esta última incluye otros campos adicionales como el usuario que lo creó, la fecha o el estado del

documento. Además almacena el *documentId,* que referencia a los datos procesados por la IA para permitir su edición en caso de modificaciones posteriores.

Los datos extraídos y generados ante la carga de un PDF son:

1. Título del documento
2. Resumen del documento
3. Autor del documento
4. Eje asociado al contenido del documento
5. País donde se emite el documento
6. Tipos de Delitos asociados al contenido del documento
7. Fuero asociado al contenido del documento

5.4.1.2. Servicio de edición de atributos

El segundo servicio consiste en la edición de los atributos de las entradas (una por cada documento) publicadas, permitiendo a los usuarios ajustar los datos generados inicialmente por la Inteligencia Artificial. Esta funcionalidad facilita la actualización del modelo con nuevos datos relevantes, optimizando la búsqueda futura dentro del sistema. Para efectuar este servicio se deben aclarar qué campos se modifican.

5.4.1.3. Servicio de búsqueda de documentos

El tercer servicio ofrecido es la Recuperación Avanzada de Información, diseñado para identificar y recuperar los documentos más relevantes en respuesta a una consulta específica realizada por un usuario en lenguaje natural. Este servicio entrega los documentos que mejor se ajustan a la consulta y también proporciona un puntaje (*score*) que evalúa qué tan bien el contenido de cada documento se alinea con lo que el usuario está buscando.

En el anexo 10.6 se encuentran las especificaciones detalladas de comunicación con los tres servicios mencionados.

5.4.2. Flujo para crear un documento en la plataforma

Para crear un documento en la plataforma, se realiza un proceso que incluye tanto el servicio de extracción de datos de documentos, como el de edición. Debajo se adjunta el flujo de interacciones:

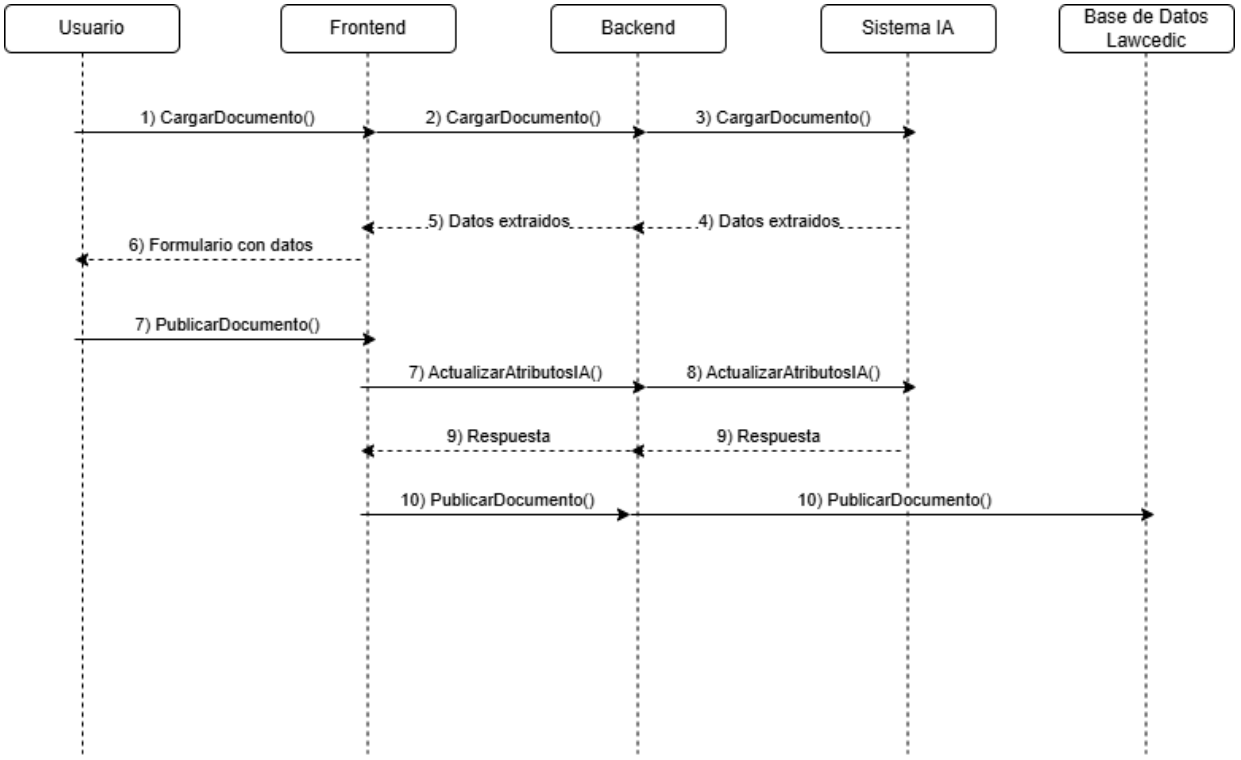

Figura 5: Diagrama de secuencia de la creación de un nuevo documento

1. Un usuario con rol author o supervisor, elige cargar un documento ya sea procesando automáticamente con IA o de forma manual.
2. El frontend de LawCEDIC procesa el PDF y genera la solicitud al backend de LawCEDIC.
3. El backend recibe la solicitud y haciendo uso de la *API-Key* que tiene almacenada de forma segura, genera una solicitud de extracción de datos de documentos a la API del sistema de IA.
4. El sistema de IA almacena el documento y, dependiendo de la solicitud, extrae todos los atributos del PDF o no. La respuesta incluirá la URL para descargar el PDF, el *documentId* externo y los atributos extraídos y resumen generado si fuera el caso.
5. El backend recibe la respuesta y reenvía al frontend.

6. El frontend ahora muestra al usuario el formulario de carga de documento, con los datos generados por la IA si fuera el caso. En esta etapa el usuario modifica y completa los campos en el formulario.
7. Una vez que el usuario decide publicar el documento en la plataforma, se genera una nueva solicitud para el sistema de IA. Se hace uso del servicio de edición de atributos enviando los campos modificados por el usuario, para que la Inteligencia Artificial tenga los datos actualizados.
8. El backend reenvía esta solicitud por medio de la *API-Key* a la API del servicio de IA.
9. El backend reenvía la respuesta de la API al frontend.
10. Si obtiene una respuesta exitosa ante el cambio, el frontend genera una solicitud final al backend para publicar el documento en la base de datos de LawCEDIC, almacenando los campos que tiene la IA y agregando otros relevantes para la plataforma.

5.4.3. Flujo para buscar documentos con el servicio de IA

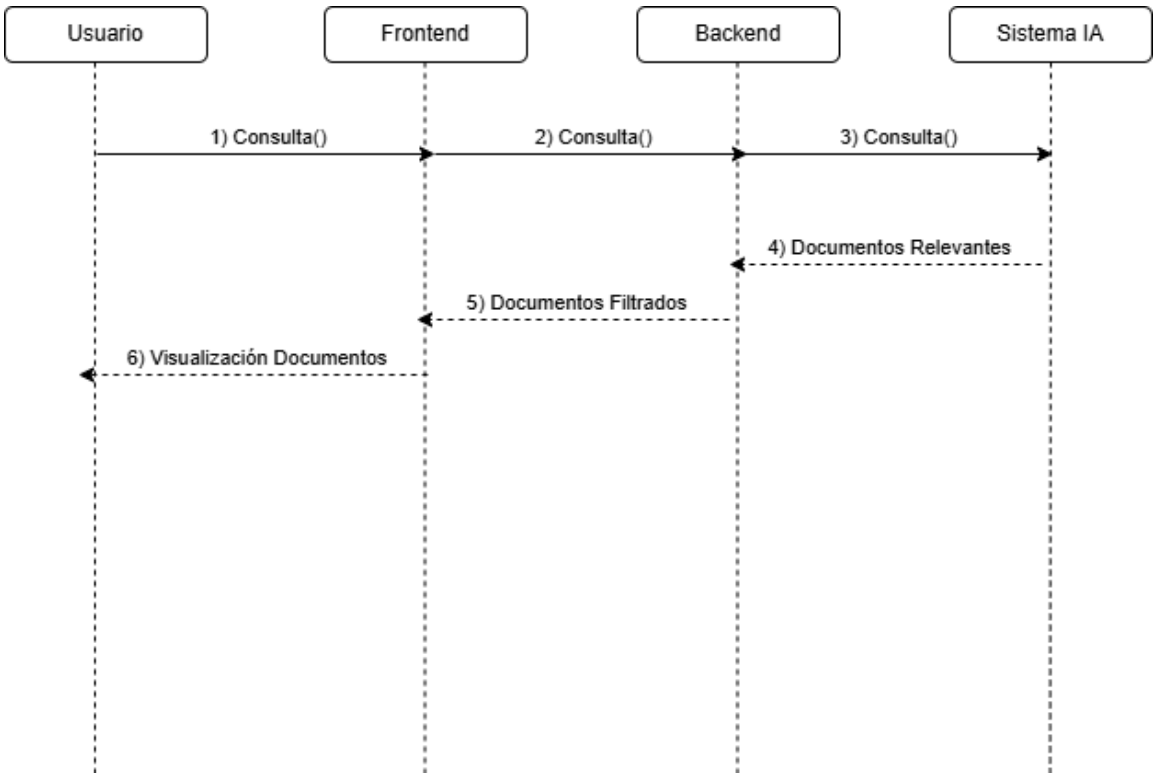

Figura 6: Diagrama de secuencia de la búsqueda de documentos con IA

Para buscar documentos en la plataforma haciendo uso del servicio brindado por la IA, se deben ejecutar una serie de pasos internos que son los mostrados a continuación:

1. Cualquier usuario dentro de la plataforma, escribe una consulta y opcionalmente agrega filtros. Esta consulta es enviada al frontend.
2. El frontend reenvía la consulta al backend.
3. El backend de LawCEDIC, haciendo uso de la *API-Key*, genera una solicitud de búsqueda de documentos al Sistema de Inteligencia Artificial.
4. El Sistema de Inteligencia Artificial retorna los documentos relevantes a esa consulta al backend junto con su *score*.
5. El backend filtra entre la base de datos propia de la plataforma aquellos que coinciden con el arreglo de *documentIds* devuelto por la API y que están en estado "Aprobado". Retorna los documentos filtrados al frontend y añade a cada uno de ellos el score que brinda la IA.
6. El frontend muestra los documentos obtenidos junto al *score*, y agrega una sección de estadísticas de la consulta.

5.5. Base de datos

El sistema de gestión de base de datos utilizada en este proyecto es MongoDB, seleccionada como parte del stack MERN debido a su simple integración con Node.js y su flexibilidad para manejar datos no estructurados. Esta característica fue esencial para gestionar los documentos judiciales, que presentan estructuras variables según el tipo (legislación, doctrina, jurisprudencia y recomendaciones, protocolos y guías). A diferencia de las bases de datos relacionales tradicionales, MongoDB permite almacenar y consultar información con esquemas heterogéneos en la misma colección, lo que simplifica el diseño y mejora la eficiencia de las consultas. Además, su escalabilidad, alta disponibilidad mediante réplicas, y capacidades avanzadas de consulta lo convierten en una opción robusta y confiable para proyectos de gran escala y datos dinámicos como este.

5.5.1. Backups

El proceso de backup es un componente esencial para garantizar la preservación y recuperación de datos críticos. El enfoque implementado combina estrategias locales y remotas para maximizar la seguridad y disponibilidad de la información.

1. **Generación del respaldo local:** El primer paso del proceso consiste en realizar un respaldo completo de la base de datos MongoDB utilizando la herramienta mongodump. Este comando extrae los datos almacenados en el contenedor de MongoDB y los guarda en un directorio temporal en la máquina virtual. Los datos se exportan en un formato estructurado que facilita su restauración futura.
2. **Transferencia remota con Rclone:** Una vez generado el respaldo local, este es transferido a un servicio de almacenamiento en la nube (Mega) utilizando la herramienta Rclone, conocida por su flexibilidad y soporte para múltiples servicios de almacenamiento. La transferencia se realiza con el comando rclone copy, que envía los archivos del respaldo local al directorio remoto especificado. Este método garantiza una conexión segura y confiable.
3. **Copia de *logs* del sistema:** Además de respaldar los datos de la base de datos, el proceso incluye la copia de los archivos de registro (*logs*) generados por la aplicación. Los *logs* son fundamentales para analizar errores, eventos y posibles incidencias en el sistema. Estos se encuentran en un directorio específico dentro de la máquina virtual y son transferidos al almacenamiento remoto en una carpeta separada.
4. **Automatización y periodicidad:** El *script* está diseñado para ejecutarse automáticamente a intervalos regulares mediante un *cron job* configurado en la máquina virtual. Esto asegura que los respaldos se realicen sin intervención manual, reduciendo riesgos.

Se destaca que ya se probó el *script* de recuperación de datos, logrando descargar el respaldo desde Mega y restaurar exitosamente la base de datos en un entorno de prueba. Esto confirma que el sistema es capaz de garantizar la restauración de los datos en caso de ser necesario.

Este enfoque garantiza una combinación efectiva de redundancia y trazabilidad. La redundancia se logra mediante la combinación de respaldo local y remoto, mientras que la organización de los directorios por timestamps permite identificar rápidamente versiones específicas.

Los scripts utilizados para el *backup* y la restauración del mismo se encuentran en la sección 9.2, en el anexo del informe.

5.6. Despliegue

El despliegue del sistema se realizó utilizando Docker, una herramienta de virtualización que permite empaquetar aplicaciones junto con todas sus dependencias en contenedores. Estos contenedores son unidades ligeras y portátiles que pueden ejecutarse en cualquier sistema que tenga Docker instalado. Esta estrategia garantiza consistencia en el comportamiento de la aplicación en todos los entornos, desde desarrollo hasta producción.

Se utilizó una metodología multicontenedor para el despliegue del sistema (frontend, backend y base de datos). Para simplificar este proceso, se utilizó Docker Compose para su orquestación. Esta herramienta permite definir y gestionar aplicaciones que constan de múltiples contenedores mediante un archivo de configuración YAML, como docker-compose.yml. De esta manera, se especifican los servicios que forman parte de la aplicación, las redes, los volúmenes y las variables de entorno necesarias para su funcionamiento. La principal ventaja de Docker Compose es la facilidad para iniciar, detener y escalar todos los servicios de la aplicación con un solo comando, lo que simplifica significativamente la administración de sistemas complejos.

El archivo .env, ubicado en el mismo directorio que el archivo docker-compose.yml, contiene las variables de entorno necesarias para configurar los tres servicios de manera segura y flexible. Docker Compose accede automáticamente a este archivo al momento de construir y ejecutar los servicios definidos en el archivo YAML. La sentencia env_file en la definición de los contenedores, especifica explícitamente que se utilicen las variables definidas en el archivo .env. Estas variables son inyectadas en el contenedor como parte de su entorno de ejecución. Este enfoque permite que las configuraciones sean portátiles y consistentes entre

entornos. Por ejemplo, las credenciales para acceder a MongoDB o a la API del modelo de IA se definen en el mismo archivo .env, facilitando la administración.

Respecto de la comunicación entre contenedores, el docker-compose.yml definido crea una red virtual interna llamada app-network, utilizando el *driver bridge*. Este driver es el modo de red predeterminado en Docker y actúa como una red interna que permite la comunicación entre los contenedores conectados a ella, manteniéndolos aislados de otros contenedores y de la red del host, salvo que se configure lo contrario. Cada contenedor conectado a esta red puede ser referenciado por su nombre de servicio definido en el archivo docker-compose.yml. Por ejemplo, el backend se comunica con MongoDB utilizando mongodb_container como *host*, en lugar de depender de direcciones IP. Esto simplifica la configuración y garantiza que los servicios puedan interactuar incluso si los contenedores se reinician.

5.6.1. Contenedores

Cada contenedor está diseñado para cumplir un propósito específico dentro de la arquitectura del sistema. El frontend gestiona la interfaz de usuario, el backend ejecuta la lógica de negocio y sirve como intermediario con la base de datos, y MongoDB almacena los datos. En conjunto, estos tres contenedores trabajan de manera eficiente, aprovechando las ventajas de Docker, Docker Compose y la red interna para ofrecer un sistema escalable, seguro y fácil de administrar.

5.6.1.1. Contenedor frontend

El contenedor del frontend aloja el servidor web Nginx, que es el encargado de servir la interfaz de usuario de la aplicación, desarrollada en React. Para asegurar que la aplicación sea accesible públicamente, el contenedor expone el puerto 5173, mapeado directamente al *host*.

Nginx no solo actúa como servidor estático, entregando los archivos generados por React, sino que también desempeña el papel de *proxy* y *proxy* inverso. Como *proxy*, gestiona solicitudes HTTP entrantes y redirige todo el tráfico HTTP hacia HTTPS, garantizando una conexión cifrada y segura. Adicionalmente, Nginx implementa un proxy inverso para

gestionar las peticiones que llegan a la ruta /api/. Estas solicitudes, en lugar de ser procesadas por el frontend, son redirigidas al contenedor backend. Este diseño protege la ubicación del backend al cliente final, aumentando la seguridad del sistema.

La configuración de Nginx incluye optimizaciones adicionales para mejorar la experiencia del usuario y la seguridad del sistema. Se establecen límites de tamaño para las solicitudes (10 MB), tiempos de espera extendidos para operaciones largas, y configuraciones específicas para manejar la carga de archivos sin interrupciones.

La imagen del contenedor frontend se construye en dos fases para optimizar tanto su tamaño como su rendimiento. En la primera fase, se utiliza una imagen base de Node.js junto con Vite, una herramienta moderna para desarrollo y construcción en React. Vite proporciona un entorno rápido y optimizado, permitiendo que las variables de entorno, definidas en el archivo de configuración de Vite, cambien dinámicamente según el modo de ejecución. Durante esta fase, se instalan las dependencias del proyecto y se realiza la construcción del mismo, generando un conjunto de archivos estáticos optimizados. En la segunda fase, los archivos generados durante la construcción se transfieren a una imagen ligera de Nginx. Este enfoque basado en múltiples fases reduce significativamente el tamaño final del contenedor y mejora su rendimiento en producción.

5.6.1.2. Contenedor backend

El contenedor del backend está diseñado para ejecutar el servidor desarrollado en Node.js. El Dockerfile utiliza una imagen base de Node.js para instalar las dependencias del proyecto en modo producción, reduciendo el tamaño del contenedor y mejorando su rendimiento.

El backend depende del contenedor de MongoDB para las operaciones de datos. Esto se logra mediante variables de entorno definidas en el archivo .env. Además, el contenedor incluye un volumen que sincroniza localmente los logs generados por la aplicación, facilitando su análisis en caso de errores o eventos inesperados. Este volumen limita el tamaño de cada archivo a 10 MB y conserva un historial de hasta cinco copias. El backend opera en el puerto 3000, mapeado directamente al puerto del host para recibir y procesar peticiones desde el frontend.

5.6.1.3. Contenedor de la base de datos

Este contenedor es responsable de almacenar los datos y mantenerlos accesibles para el backend a través de la red app-network. Para garantizar la persistencia de los datos, se utiliza un volumen local que sincroniza los datos almacenados en el contenedor con un directorio en el sistema de archivos del host. Esto asegura que la información no se pierda incluso si el contenedor es eliminado o reiniciado. El contenedor expone el puerto 27017, necesario para la comunicación con el backend y para tareas de administración local.

5.6.2. Elección del tipo de solución (VPS)

La elección del entorno de despliegue fue un aspecto clave para garantizar el rendimiento y la flexibilidad del proyecto. Al inicio del desarrollo, se evaluaron las siguientes alternativas:

- **Solución a nivel plataforma (PaaS):** Una opción considerada fue la contratación de un servicio de hosting basado en PaaS, como AWS Elastic Beanstalk, Google Firebase o Microsoft Azure. Esta solución permite delegar gran parte de la configuración y el mantenimiento en el proveedor, simplificando el proceso de despliegue. Sin embargo, el uso de PaaS tiene limitaciones significativas en términos de flexibilidad y control. Los recursos del servidor son compartidos con otros clientes del proveedor, lo que podría impactar negativamente en el rendimiento de la plataforma en caso de que otro cliente consuma una cantidad desproporcionada de recursos. Además, cualquier problema técnico depende directamente del soporte del proveedor, lo que puede generar tiempos de espera largos para resolver problemas menores. Esto impacta negativamente en la disponibilidad, ya que el servicio puede quedar inactivo hasta recibir asistencia, y en la mantenibilidad, porque la dependencia del proveedor dificulta la resolución ágil de incidencias.
- **Servidor de la Universidad Austral:** Otra posibilidad era utilizar una máquina virtual proporcionada por la Universidad Austral. Aunque esta opción garantiza un control sobre la administración, también requiere una comunicación constante y fluida con los administradores del servidor. Por otro lado, por parte de OCEDIC preferían optar por otro proveedor, ya que consideraban que esta elección, junto con el proceso de

contratación y futuros contactos, no sería ágil debido a la burocracia de la Universidad.

- **Solución a nivel infraestructura (IaaS):** Por último, se consideró contratar un servidor virtual en un centro de datos basado en IaaS, como AWS EC2, Microsoft Azure Virtual Machines o un proveedor local. Esta solución ofrece control total sobre la configuración del servidor y permite optimizar los recursos y el rendimiento de acuerdo a las necesidades del sistema. Al tener acceso administrativo completo, es posible personalizar el entorno de ejecución, configurar medidas de seguridad avanzadas y escalar el sistema según sea requerido.

Tras un análisis de las alternativas, se optó por implementar la aplicación en una máquina virtual (VPS) proporcionada por Ecolan, un cooperativa local ubicada en Batán, que cuenta con un centro de datos.

La solución ofrecida por Ecolan destacó frente a alternativas como el uso de servicios PaaS (AWS Elastic Beanstalk o Google Firebase) u otros servidores locales. Los servicios de Ecolan, basados en IaaS, proporcionan acceso completo a la administración del servidor. Además, la ubicación geográfica asegura baja latencia para los usuarios nacionales y latinoamericanos (público objetivo del proyecto) mejorando su experiencia.

Otro factor determinante fue la buena relación que el director del proyecto, Hernán Hinojal, mantiene con los administradores de Ecolan, lo que simplificó la comunicación inicial, y garantiza un diálogo directo ante cualquier eventualidad o necesidad técnica. Además, el costo competitivo de aproximadamente $41,000 hace que esta opción sea más accesible comparada con servicios internacionales como AWS o Azure, que podrían requerir una inversión mayor para obtener características similares.

Aunque esta opción requiere mayor conocimiento técnico para configurar y mantener el servidor, sus beneficios en términos de rendimiento y escalabilidad la convierten en la solución más adecuada para el desarrollo de esta aplicación.

La máquina virtual que recibimos tiene instalado Debian 12, una distribución de Linux reconocida por su estabilidad, seguridad y soporte a largo plazo. Esta base proporcionó un entorno robusto y confiable para el despliegue de la aplicación. Además, durante los

primeros meses, Ecolan ofreció el uso gratuito de la máquina virtual, lo que permitió al equipo probar y configurar el sistema sin incurrir en costos adicionales. Este nivel de flexibilidad y apoyo inicial hubiera sido difícil de obtener con otros proveedores internacionales, lo que refuerza la ventaja de esta elección.

5.6.3. Dominio web

El dominio web es un componente esencial de la infraestructura del sistema, ya que facilita el acceso de los usuarios a la plataforma. En este caso, se aprovechó el dominio principal del observatorio, ocedic.com, que ya estaba configurado para alojar la página institucional gestionada mediante Wordpress.

Para la plataforma, se creó el subdominio lawcedic.ocedic.com, gestionado por la empresa DataWebHosting, encargada de la administración del dominio principal. El subdominio fue solicitado mediante comunicación por correo electrónico con el soporte técnico de la empresa, quienes configuraron todo el proceso de manera eficiente y sin necesidad de intervenciones adicionales por parte del equipo técnico.

El uso del subdominio presentó ventajas importantes, ya que eliminó la necesidad de adquirir un dominio nuevo, lo que redujo los costos asociados. Asimismo, al mantener todos los servicios bajo el dominio principal del observatorio, se logró una integración más coherente entre la página institucional y la nueva plataforma. Esto mejora la experiencia de los usuarios y refuerza la identidad del observatorio, ofreciendo un acceso unificado y profesional.

Gracias a esta configuración, se logró obtener una dirección confiable y fácil de recordar para los usuarios a través de lawcedic.ocedic.com.

5.6.4. Seguridad

La seguridad de LawCEDIC se aborda desde múltiples frentes, con un enfoque especial en proteger la configuración de la máquina virtual donde está desplegada la app. Las medidas incluyen la configuración de certificados SSL, HTTPS, firewall, Fail2ban y el uso de proxies (directos e inversos) para gestionar el tráfico y mitigar amenazas.

- **Certificados SSL y HTTPS:** La implementación de certificados SSL asegura que todas las comunicaciones entre el cliente y el servidor estén cifradas. Esto se logra mediante el protocolo HTTPS, que protege los datos contra accesos no autorizados durante la transmisión.

  En este caso, el certificado SSL fue configurado utilizando Let's Encrypt, un proveedor de certificados gratuitos y automatizados. El certificado se instala en el servidor Nginx del contenedor frontend. Además, se configura la redirección automática de HTTP a HTTPS para garantizar que todas las solicitudes utilicen una conexión segura. El cifrado asegura que los datos transferidos entre el cliente y el servidor no puedan ser interceptados o manipulados. Además genera confianza en los usuarios al navegar por el sitio, a causa del candado en el navegador.

- **Firewall con UFW:** El firewall UFW (Uncomplicated Firewall) protege la máquina virtual controlando qué conexiones entrantes y salientes están permitidas. La configuración actual bloquea por defecto todas las conexiones entrantes, excepto aquellas explícitamente permitidas, lo que reduce la superficie de ataque.

  Las reglas definidas permiten el acceso a puertos esenciales como:

  - 22 y 2222: Para conexiones SSH.
  - 80 y 443: Para tráfico HTTP y HTTPS.
  - 3000: Para el backend.
  - 10050: Servicios adicionales necesarios.

  La estrategia de permitir únicamente los puertos necesarios minimiza el riesgo de accesos no autorizados. Además, el registro de actividad está habilitado para monitorear intentos de conexión sospechosos o no autorizados.

- **Fail2ban:** Fail2ban es una herramienta que protege al sistema contra ataques de fuerza bruta y otras amenazas al analizar los logs del sistema y bloquear direcciones IP que presentan actividad maliciosa. Fail2ban está configurado con múltiples *jails* específicos, como:

- sshd: Protege las conexiones SSH monitoreando intentos de inicio de sesión fallidos.
- nginx-badbots y nginx-botsearch: Detecta bots maliciosos que intentan explotar vulnerabilidades en el servidor web.
- nginx-noscript: Previene ataques basados en scripts maliciosos.
- mysql-auth y nginx-http-auth: Protege los servicios de autenticación en MySQL y Nginx.
- postfix: Mitiga posibles ataques al servidor de correo.

Fail2ban se integra con los logs generados por los contenedores del sistema, monitoreando la ruta /var/log/nginx/access.log. Esto permite estar al tanto de ataques o accesos no autorizados que puedan dirigirse específicamente al backend o la base de datos.

- **Uso de proxies:** El uso de *proxies* y *proxies* inversos en la arquitectura del sistema cumple una función crítica, ya que permite gestionar el tráfico, filtrar solicitudes y garantizar una capa adicional de seguridad. Estas configuraciones aseguran que tanto el frontend como el backend estén protegidos y optimizados para manejar de manera eficiente las solicitudes entrantes.
    - **Proxy en el Frontend:** El *proxy* configurado en Nginx para el frontend filtra y controla las solicitudes entrantes. Este *proxy* aplica restricciones importantes para proteger el sistema, como un límite de tamaño para las solicitudes de 10 MB (client_max_body_size 10M), rechazando aquellas que superen este umbral para prevenir ataques de denegación de servicio (DoS). También se configuraron tiempos de espera extendidos para la carga de documentos de gran tamaño, evitando desconexiones repentinas con parámetros como client_body_timeout y send_timeout.

      Nginx también está configurado para redirigir automáticamente todo el tráfico HTTP a HTTPS, asegurando que todas las conexiones entre el cliente y el servidor estén cifradas y protegidas contra posibles ataques de interceptación.

- Proxy Inver**so en el Backend:** El *proxy* inverso configurado en Nginx para el backend actúa como un filtro y un intermediario que encapsula el servidor Node.js. Todas las solicitudes que llegan desde el frontend al backend pasan primero por Nginx, donde se validan antes de ser redirigidas. Se redirige entonces cualquier petición que comience con la ruta /api/ hacia el backend. Esta redirección garantiza que el cliente no tenga acceso directo al backend, manteniendo su ubicación y lógica protegidas de manera efectiva.

  Nginx también agrega encabezados específicos a las solicitudes, como X-Real-IP y X-Forwarded-For, lo que permite registrar la IP del cliente original y rastrear actividades sospechosas en los registros del sistema. Este enfoque fortalece la seguridad al permitir monitorear y analizar el tráfico en busca de patrones fuera de lo común.

5.6.5. Despliegue automatizado con GitHub Actions

El despliegue de la plataforma fue configurado para realizarse de manera automática, haciendo uso de herramientas de integración continua (CI/CD). Gracias a este enfoque, cada actualización del sistema se implementa automáticamente en el entorno de producción. Esto no solo elimina la necesidad de intervención manual, sino que también optimiza la estabilidad del sistema. Al depender de flujos de trabajo predefinidos, se garantiza que cada versión de la plataforma tenga que pasar por los mismos procesos de construcción, prueba y despliegue, evitando errores humanos y asegurando coherencia en el entorno de producción.

5.6.5.1. Herramientas utilizadas y estructura del proyecto

Para lograrlo se utiliza GitHub Actions, una herramienta de automatización integrada en GitHub, que permite ejecutar tareas en respuesta a eventos dentro de un repositorio. Mediante flujos de trabajo definidos en archivos YAML, se especifican acciones como la compilación, prueba y despliegue del código.

Para comprender cómo funciona el proceso de despliegue es necesario conocer los repositorios de Github que conforman el proyecto:

- **Frontend-LawCEDIC**: contiene el código del frontend desarrollado en React y compilado con Vite.
- **API-LawCEDIC**: contiene el backend desarrollado en Node.js con Express.
- **Produccion-LawCEDIC**: gestiona el despliegue. Este repositorio no contiene código fuente, sino que cuenta únicamente con un script de GitHub Actions que se activa cuando alguno de los otros dos repositorios realizan cambios en la rama de producción.

El repositorio del frontend y el del backend también cuentan con scripts, los cuales detectan los cambios y disparan el workflow principal en Produccion-LawCEDIC. Además, cada repositorio cuenta con dos ramas principales:

- **main**, utilizada para desarrollo y pruebas.
- **production**, almacena el código estable y se despliega en producción.

Es importante destacar el uso de Docker Hub, una plataforma que facilita la gestión y distribución de imágenes Docker. La principal ventaja de utilizarla es que la máquina virtual no necesita contener el código fuente del proyecto, sino que simplemente descarga las imágenes ya compiladas y listas para ejecutarse. Esto mejora la seguridad al reducir la exposición del código, evita manipulaciones accidentales del mismo y garantiza también que todos los entornos de despliegue sean consistentes.

Explicado de manera breve, el funcionamiento es el siguiente: cada vez que se realiza un push a la rama production en el frontend o backend, el script pertinente de GitHub Actions activa el proceso de despliegue en Produccion-LawCEDIC, el cual garantiza que los cambios lleguen automáticamente a la máquina virtual de producción. El proceso paso a paso se explica a continuación.

5.6.5.2. Proceso de despliegue automatizado

1. **Activación del despliegue:** Cada vez que se hace un push a la rama production de Frontend-LawCEDIC o API-LawCEDIC, se ejecuta un workflow de GitHub Actions. Este workflow detecta cambios en la rama production y envía una notificación a Produccion-LawCEDIC para que inicie el despliegue.

2. **Despliegue en producción:** Cuando Produccion-LawCEDIC recibe la señal de despliegue, su workflow realiza una serie de pasos automatizados. Estas tareas se realizan remotamente, en los servidores de Github.
    a. Paso 1: Clonación del Código: dependiendo de si se está desplegando el frontend o el backend, se clona la última versión del código desde la rama production.
    b. Paso 2: Construcción y Subida de la Imagen Docker: se construye una imagen Docker actualizada con el código más reciente y se sube a Docker Hub.
    c. Paso 3: Conexión con la Máquina Virtual: se establece una conexión segura SSH con la máquina virtual de Ecolan, mediante la cual el script gana acceso a la misma.
    d. Paso 4: Despliegue en la Máquina Virtual: se descargan las nuevas imágenes Docker desde Docker Hub y se corre el docker-compose.yml del sistema de archivos de la máquina virtual, lo cual ejecuta estos nuevos contenedores.

La interacción entre los distintos servidores se puede ver de manera más clara en el siguiente diagrama.

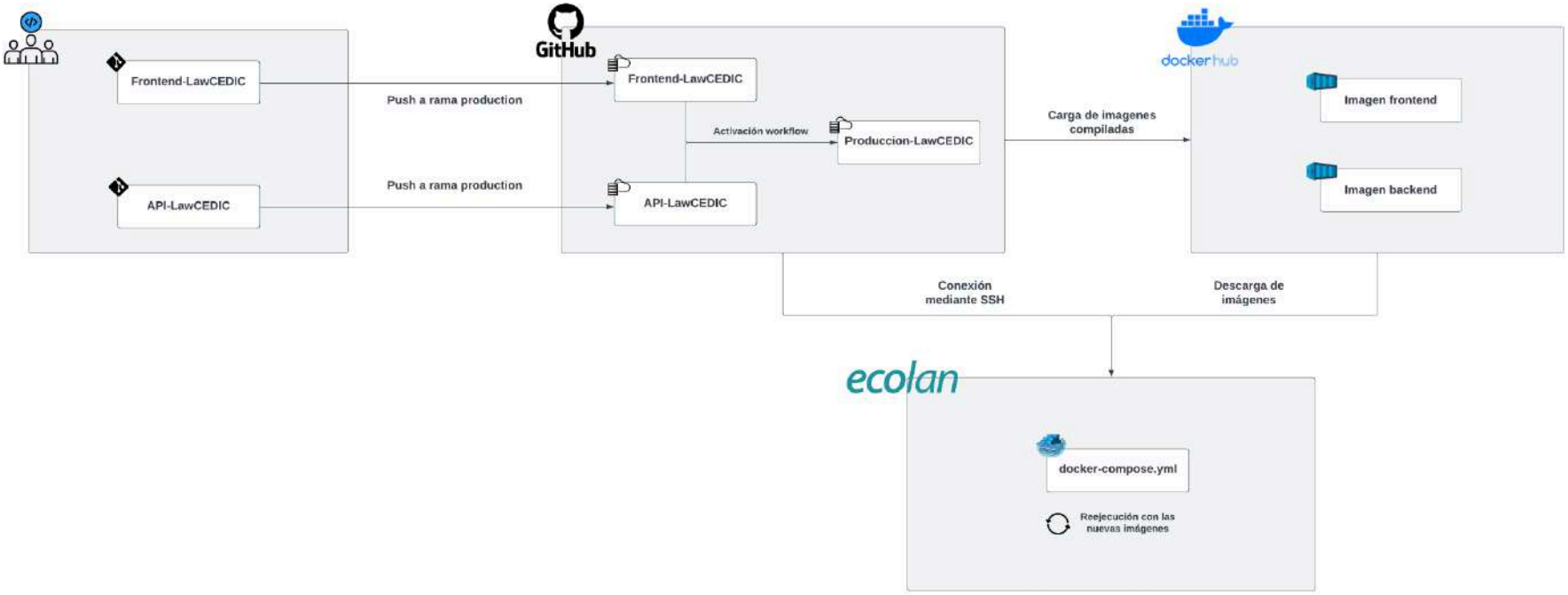

Figura 7: Interacción entre los servicios que participan del despliegue de LawCEDIC

Esta metodología de despliegue no se tuvo en cuenta durante la planificación inicial para la estimación de tiempos. Sin embargo, a medida que se avanzaba con la configuración de la máquina virtual, el equipo se dio cuenta de que automatizar este proceso traería grandes mejoras a futuro: a medida que la plataforma crece, el mismo flujo puede adaptarse para incluir nuevas pruebas automatizadas, balanceo de carga o incluso despliegues en múltiples entornos. En resumen, automatizar el despliegue no solo facilita el proceso, sino que también ayuda a que la plataforma crezca y se mantenga estable a largo plazo.

El código con la implementación del proceso de despliegue se encuentra en la sección anexo, donde se incluyen los scripts de Github Actions de cada repositorio.

6. Producto

6.1. Producto obtenido

El producto resultante del diseño y desarrollo es una plataforma web compuesta por una interfaz de usuario accesible en lawcedic.ocedic.com y una API que atiende las peticiones enviadas desde la página y se comunica con la base de datos para almacenar o recuperar información. Además, la plataforma integra un servicio de inteligencia artificial provisto por Augmented Experiences, consultora de META, para optimizar la carga y búsqueda de documentos. A continuación se enumeran las características de la plataforma a partir de las vistas, que dependen del rol del usuario.

- ***Landing page* para iniciar sesión**: simple formulario de inicio de sesión. Esta página es visible únicamente para clientes que aún no tienen una sesión activa. Puede observarse un botón de recuperación de contraseña que redirige al usuario a una página en donde debe ingresar su dirección de correo y, si tiene una cuenta, se le enviará un email con un *link* para actualizar la contraseña.

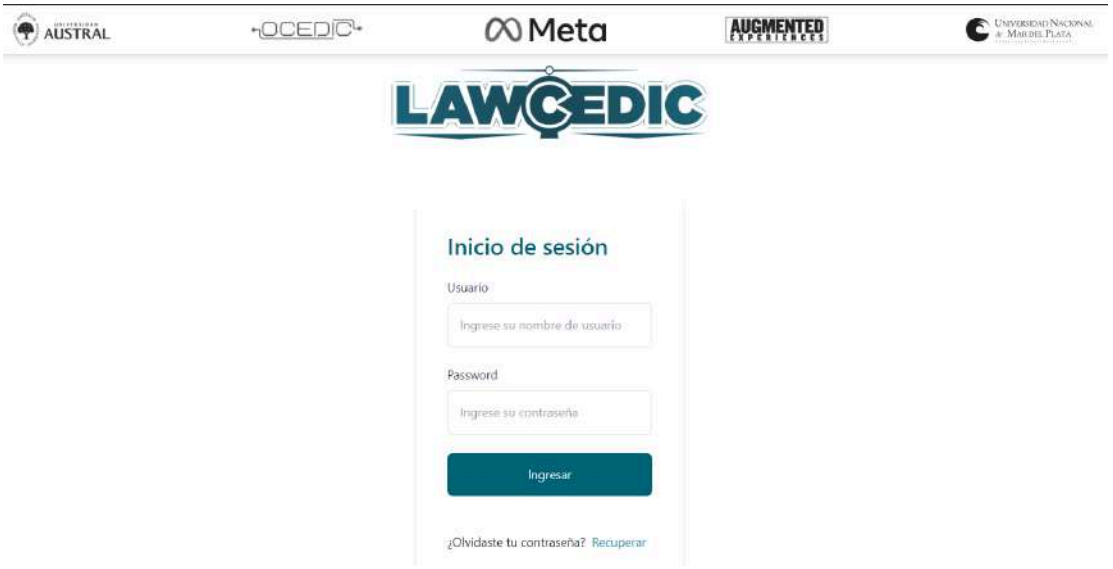

Figura 8: Landing page de LawCEDIC con formulario de inicio de sesión

- ***Home* (cualquier usuario)**: Una vez iniciada la sesión, el usuario se encuentra con un buscador, mediante el cual puede hacer búsquedas de documentos judiciales a partir de un texto en lenguaje natural. Además, a través de los botones ubicados en el centro, cuyos nombres corresponden a los ejes temáticos del Observatorio, pueden realizarse búsquedas manuales a través de esos filtros. Debajo de la barra de búsqueda hay un recuadro con los últimos documentos cargados. A la izquierda de la pantalla se tiene una barra de navegación que permite navegar a otras secciones. Dependiendo del rol del usuario, estas funcionalidades serán limitadas. En la captura de pantalla a continuación, se ven la gran mayoría de las secciones dado que se inició sesión con un usuario administrador (a excepción de la sección del formulario, a la cual no tiene acceso).

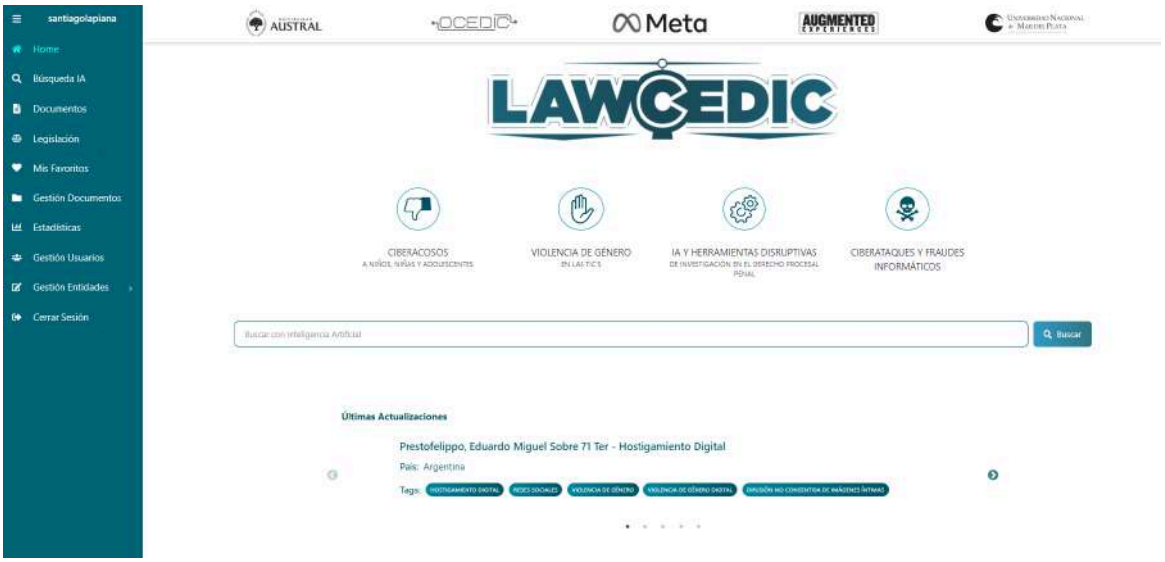

Figura 9: Home de LawCEDIC

- **Búsqueda IA (cualquier usuario)**: Luego de ingresar la consulta en lenguaje natural para que sea procesada por la inteligencia artificial, a cada documento se le muestra un score, que indica la coincidencia entre lo buscado y lo encontrado. En la siguiente captura de pantalla se buscó “cuestión de competencia sobre dispositivos electrónicos” y el primer documento relevante tiene un puntaje de similaridad del 85% mientras que el último, un 75%.

Figura 10: Resultados encontrados por la búsqueda con IA de LawCEDIC

- **Documentos (cualquier usuario)**: Se tiene un filtrador manual que permite buscar documentos mediante la selección de criterios como tipo de documento, país, tipo de delito, eje temático y etiqueta. Los documentos recuperados se ven a la derecha y en caso de ser necesario se puede desplazar hacia abajo para ver los demás. Por cada uno de ellos se ve el título, la fecha, las características propias de ese tipo (por ejemplo fuero, expediente y tribunal en caso de jurisprudencia), etiquetas asociadas (llamadas “voces”), un fragmento del resumen del contenido y finalmente un botón que permite ver toda la información completa. Al ingresar al documento de forma individual, el usuario tiene también la opción de descargar el archivo completo en formato PDF.

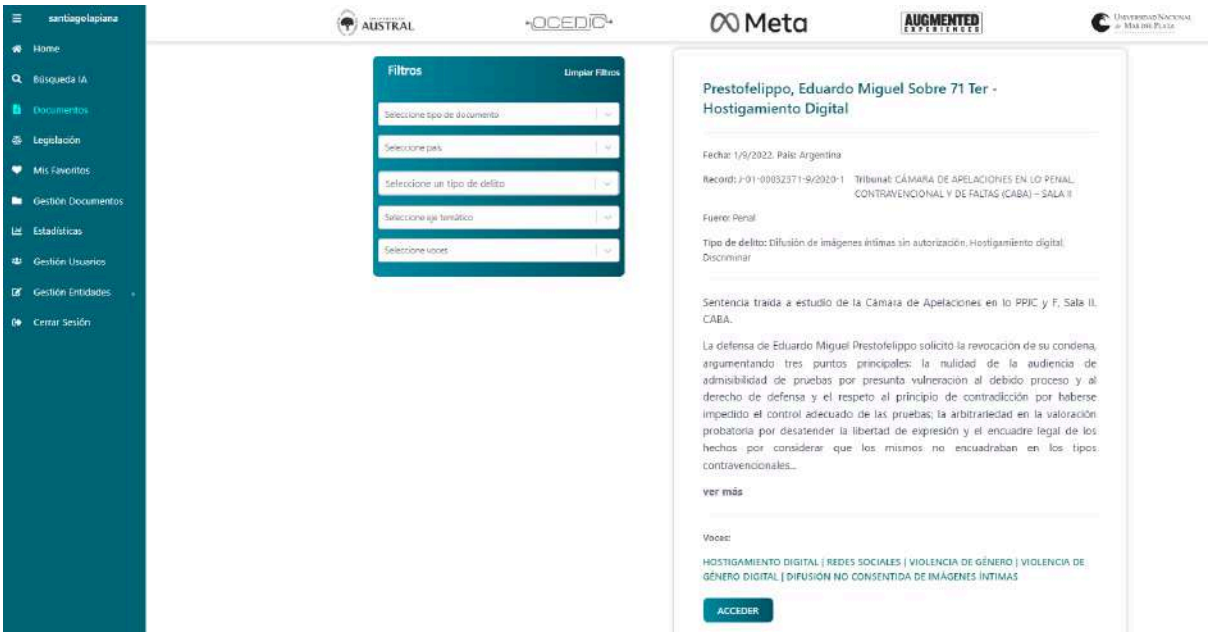

Figura 11: Sección de documentos de LawCEDIC con formulario de filtros manuales

- **Legislación (cualquier usuario)**: Permite la visualización de los árboles de legislación. En principio se encuentran los esquemas normativos de primer nivel, llamados tipos de norma. Luego, al seleccionar uno se despliegan sus hijos (por ejemplo, al seleccionar “Códigos Nacionales” se pueden encontrar “Código Penal - Argentina”, “Código Civil y Comercial - Argentina”, “Código Penal - Chile”, etc). Los esquemas normativos de segundo nivel siempre muestran el país al cual pertenecen. En consiguiente, se estructuran las legislaciones en libros, capítulos o secciones según la norma y finalmente se encuentran los artículos que al seleccionarlos despliegan su contenido.

Figura 12: Buscador de legislación a través de esquemas normativos de LawCEDIC

- **Favoritos (cualquier usuario)**: En esta sección, cada usuario puede encontrar sus documentos marcados como favoritos y organizarlos en carpetas para una mejor gestión. Siempre que consulte un documento en cualquiera de las formas mencionadas, al colocar el cursor sobre él, aparecerá un signo '+' que permite agregarlo a sus favoritos y crear o seleccionar una carpeta para guardarlo. De esta manera, se facilita el acceso a documentos previamente vistos y considerados relevantes.

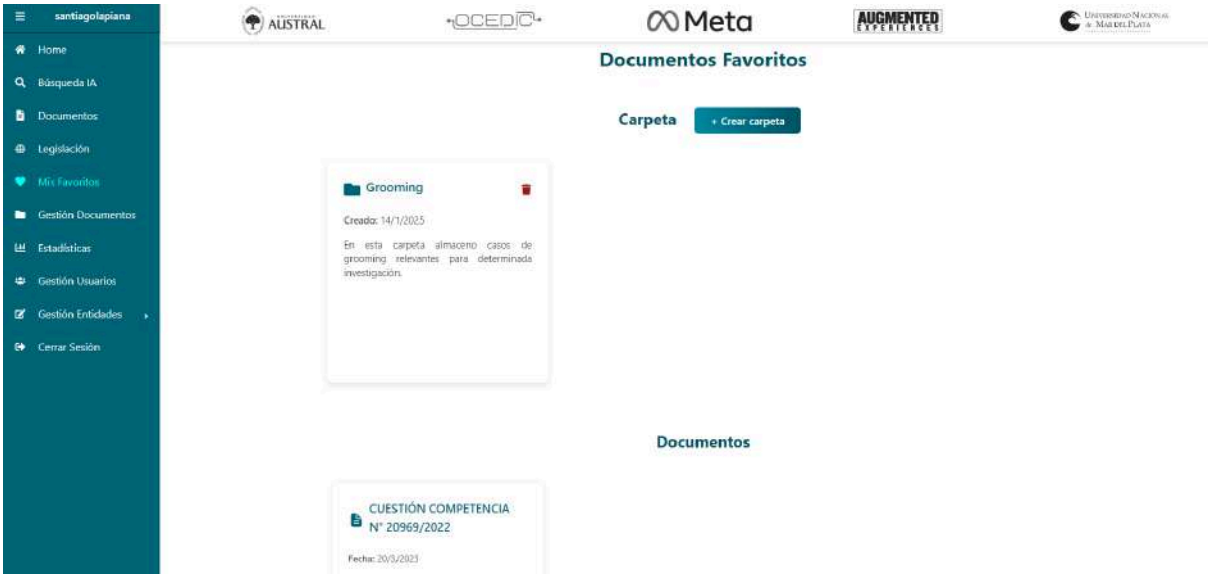

Figura 13: Sección de documentos favoritos de LawCEDIC

- **Gestión de documentos (administradores y supervisores)**: Permite ver todos los documentos cargados, junto con la información de quién los subió y cuándo. Los administradores y supervisores pueden aprobar o dejar en pendiente cada documento. Solo los documentos aprobados estarán disponibles para su visualización y búsqueda; en caso contrario, permanecerán en una cola de espera hasta su revisión.

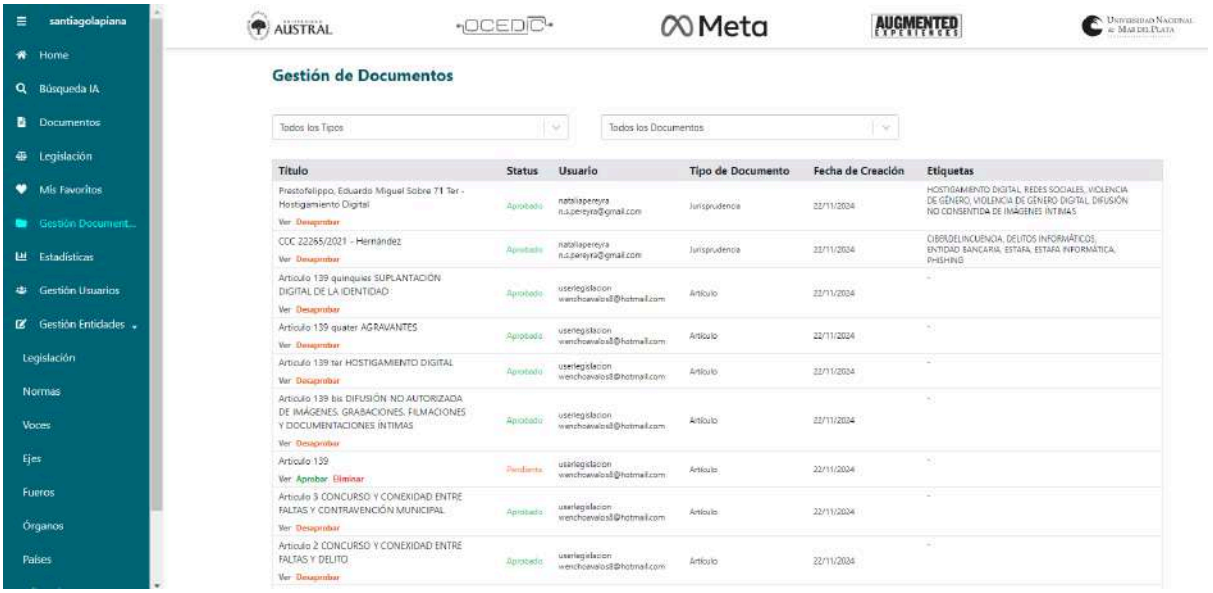

Figura 14: Sección de gestión de documentos de LawCEDIC

- **Gestión de entidades (administradores y supervisores)**: Permite crear esquemas normativos, etiquetas, ejes, fueros, órganos o países que luego serán seleccionados por los usuarios de carga o sugeridos por la IA a la hora de cargar los documentos.

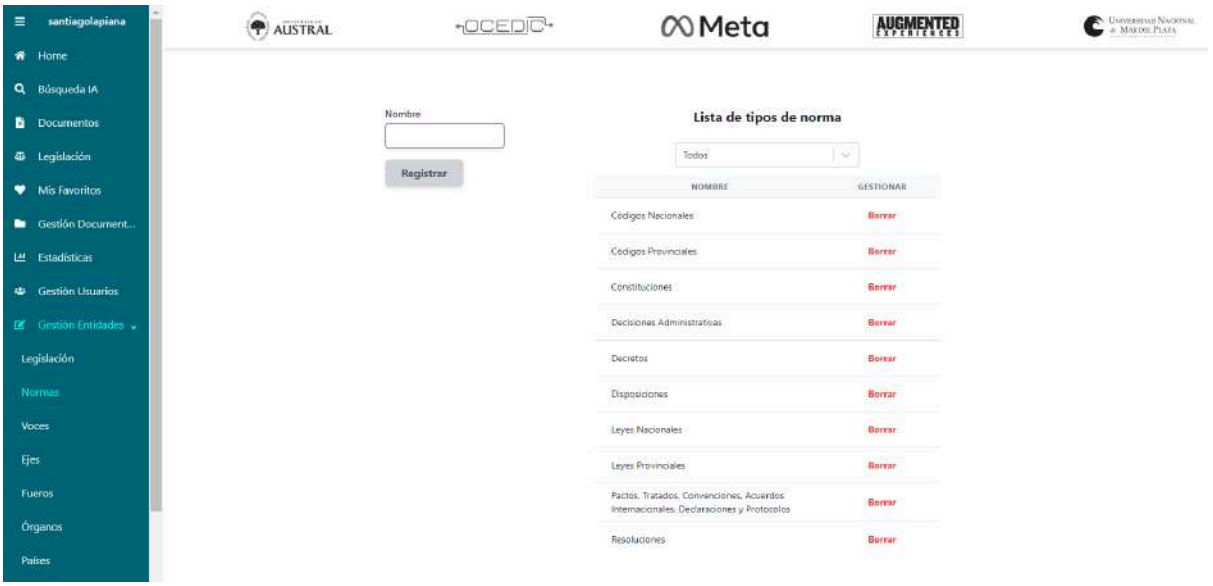

Figura 15: Vista de gestión de tipos de norma de LawCEDIC

- **Estadísticas (administradores y supervisores)**: Muestra información relevante para quienes se encargan de la supervisión del sistema, como la cantidad de documentos aprobados contra los pendientes, la cantidad de cada tipo de documento y eje temático y cuántos documentos cargó cada autor.

Figura 16: Estadísticas de documentos cargados de LawCEDIC

- **Formulario de carga (supervisores y autores)**: Es la funcionalidad de carga de documentos al sistema. Se carga un archivo en formato PDF, se selecciona el tipo y luego se procede a la carga manual o al procesamiento con IA. El uso del modelo permite una detección automática de las características del documento lo cual agiliza el trabajo de publicación.

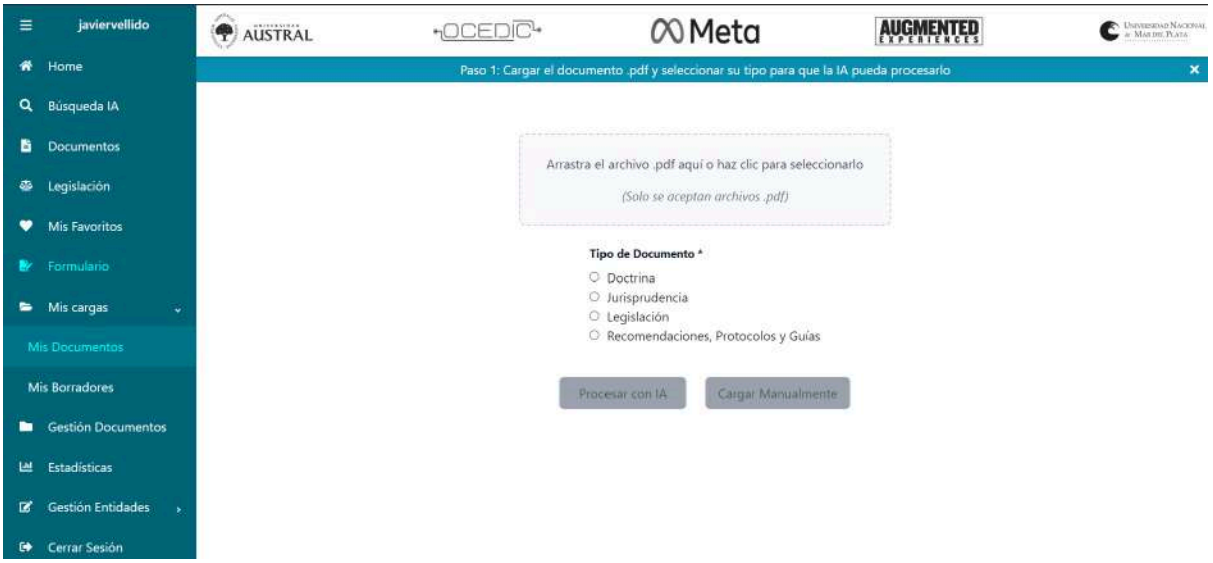

Figura 17: Vista de carga de un nuevo documento de LawCEDIC

- **Gestión de usuarios (solo administradores)**: Permite crear nuevos usuarios, cambiar los roles de los existentes y desactivar o restaurar cuentas.

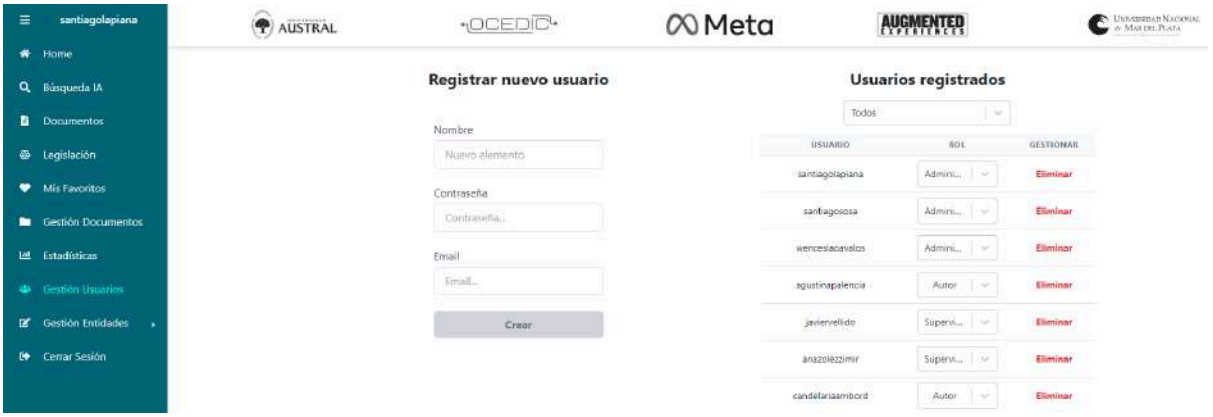

Figura 18: Gestión de usuarios de LawCEDIC

7. Memoria del proyecto

7.1. Cumplimiento de los objetivos

Todos y cada uno de los requerimientos fueron satisfechos. Se logró con éxito el objetivo de implementar una plataforma que permita gestionar documentos judiciales de manera eficiente, facilitando el trabajo de abogados y analistas de casos, con un proceso de carga ampliamente automatizado. Se establecieron las bases para que en un futuro cercano OCEDIC pueda comercializar con suscripciones al software y se documentaron adecuadamente las cuestiones técnicas, facilitando la incorporación de nuevas funcionalidades. Se alcanzó un nivel de robustez y escalabilidad ampliamente superiores a los posibles con un sistema de gestión de contenidos y paquetes de terceros como lo es Wordpress, que era el curso de acción inicial, previo a la definición de este proyecto.

7.2. Conformidad de los usuarios

Si bien los usuarios finales externos al observatorio aún no han tenido la oportunidad de probar la plataforma, ya que su lanzamiento oficial está previsto para finales del primer cuatrimestre de 2025, la validación se llevó a cabo de manera continua con el referente funcional y su equipo, conformado por abogados y analistas especializados en cibercrimen, evidencia digital e investigaciones criminales.

Por su parte, los usuarios encargados de cargar documentos en la plataforma han visto simplificada gran parte de su trabajo gracias al resumen automático generado por inteligencia artificial y las sugerencias de etiquetas. Con la nueva plataforma, el proceso de carga de documentos, que abarca la lectura, comprensión y carga del formulario se ha optimizado significativamente. Antes se requería un mínimo de 1 hora y 15 minutos, que podía extenderse varias horas dependiendo de la longitud y el tipo de documento, mientras que ahora puede resolverse en aproximadamente 30 minutos, lo que representa un ahorro del 60% en el tiempo empleado. Actualmente, la mayor parte del esfuerzo se concentra en la lectura y análisis del documento, ya que la elaboración del resumen y la asignación de etiquetas se han automatizado.

Además de la automatización, la nueva interfaz y sus facilidades han mejorado la predisposición de los usuarios de carga. Como resultado, los curadores de documentos han comentado que vieron optimizado su flujo de trabajo y que su productividad aumentó.

7.3. Implementación de metodologías

Inicialmente se realizó un documento borrador compartido por los tres desarrolladores en el cual se establecieron las entidades de la base de datos y las interfaces entre frontend y backend para poder paralelizar las tareas. Esta tarea se realizó en forma conjunta y luego se asignó un desarrollador a la implementación de la API del backend y los dos desarrolladores restantes se enfocaron en todas las vistas y funcionalidades del frontend. Este documento borrador hizo sumamente eficiente la comunicación y el proceso de codificación.

Luego de asignar y realizar cada tarea las premisas eran actualizar el tablero Trello y la bitácora con su detalle y el tiempo que llevó, pero la realidad es que no se cumplió en todos los casos dado que en varias ocasiones los horarios de trabajo en el proyecto eran irregulares, no constantes y en algunos casos se intercalaban con horarios de cursada, laborales o con semanas de entregas y parciales. No obstante, se dejaron registradas la mayoría de las tareas realizadas, lo que facilitó su comparación con lo planificado.

7.4. Comparación entre planificación esperada y ejecutada

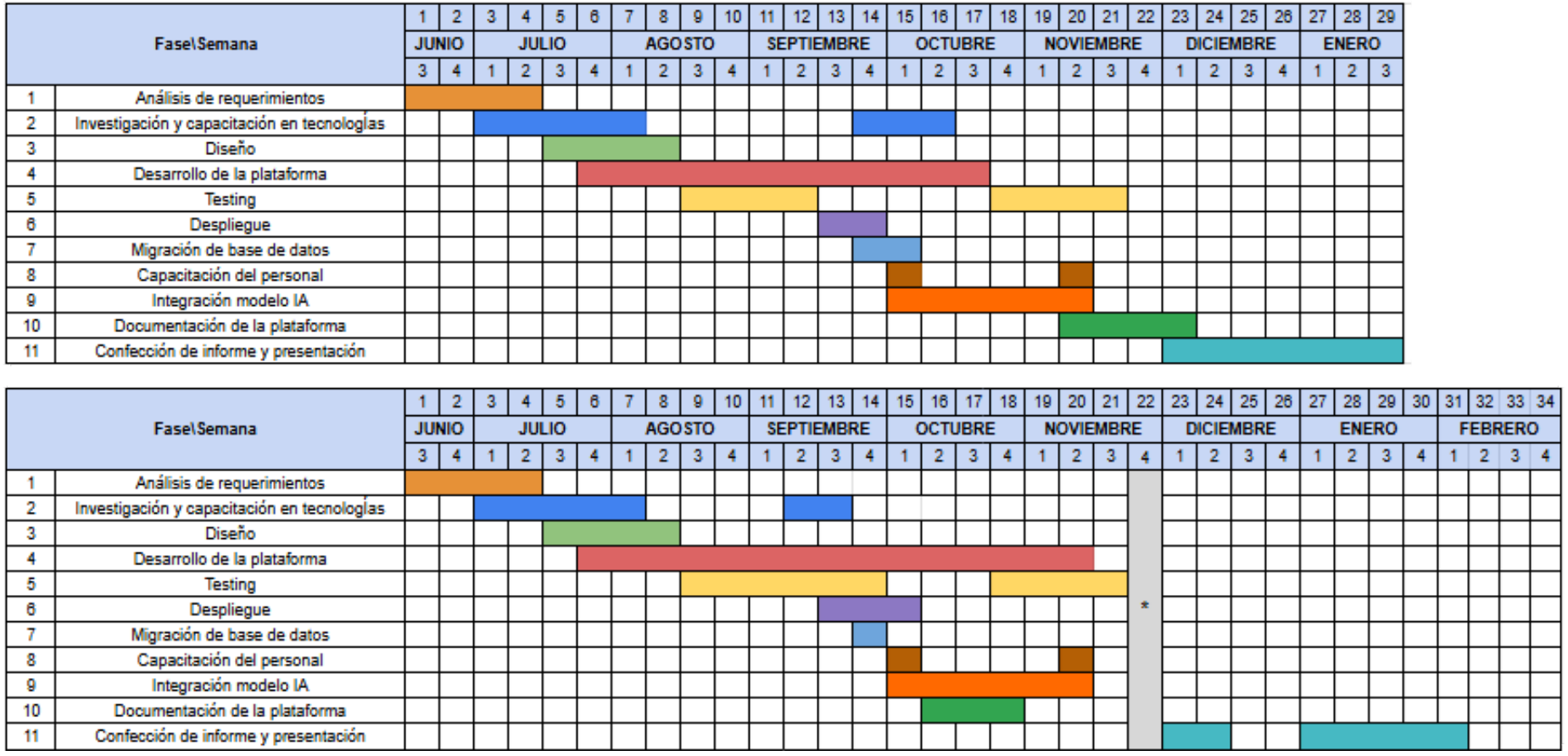

*Figura 19: Comparación de diagramas de Gantt entre planificación inicial y lo ejecutado (*Viaje a San Luis para participar en el 4to Encuentro Federal de Cibercrimen y Nuevas Tecnologías)*

Antes de realizar una comparación detallada tarea a tarea de lo planificado y lo ejecutado, es necesario reconocer un importante error en la gestión del proyecto: se asumió erróneamente que las horas de trabajo de todas las semanas se mantendrían constantes. Esto es irreal dado que cada estudiante tenía sus responsabilidades externas a este proyecto sumado a que en las semanas de las fiestas la productividad se ve reducida significativamente. Sin embargo, capitalizar este tipo de errores en términos de aprendizaje resulta fundamental para el crecimiento como profesionales.

La ausencia de las secciones de comparación para las tareas de análisis de requerimientos y diseño se debe a que, al momento de realizar la planificación inicial detallada con la estimación de plazos, dichas tareas ya se habían finalizado. Por lo tanto, simplemente se registró el tiempo que habían tomado.

- **Investigación y capacitación en tecnologías:** Desde la planificación del proyecto, se estableció que la definición del stack tecnológico debía basarse en la experiencia previa del equipo y en la compatibilidad con la solución a desarrollar. Con el documento de requerimientos ya definido, se realizó además una investigación de las dependencias y librerías necesarias para implementar las funcionalidades previstas. Además, en una fase más avanzada del proyecto, se llevó a cabo un estudio de las opciones de hosting disponibles. Se evaluaron diversas soluciones en función de su costo, rendimiento, flexibilidad y seguridad. A pesar de contar con una base de conocimientos sobre virtualización, el equipo requirió capacitación adicional en áreas específicas no contempladas en la planificación original. En particular, fue necesario adquirir experiencia en despliegues automáticos mediante GitHub Actions. Asimismo, se profundizó en medidas de seguridad para entornos en producción. Si bien esta capacitación no alteró significativamente los tiempos generales del proyecto, sí representó una inversión adicional de horas en investigación y pruebas.

- **Desarrollo**: El desarrollo representó la etapa central del proyecto, abarcando la implementación del backend, frontend y base de datos. En la planificación original, se estableció una duración estimada de 13 semanas.

En términos de tiempos de desarrollo, la ejecución fue en gran medida acorde a lo planificado, cumpliendo con la implementación de las funcionalidades dentro del margen de semanas previsto. Sin embargo, la distribución de esta fase se extendió por tres semanas adicionales, alcanzando la tercera semana de noviembre. Esta extensión no se debió a dificultades técnicas, sino a la incorporación de mejoras en la interfaz del frontend. Estos ajustes, aunque no alteraron la funcionalidad de la plataforma, fueron incorporados para optimizar la presentación visual del sistema, especialmente de cara a su exposición en el Congreso de San Luis. Estos cambios menores fueron orientados a garantizar que la plataforma tuviera una apariencia más atractiva y profesional en su primera presentación oficial.

- ***Testing***: En cuanto al *testing* automático de la API, la mayor parte de los *tests* se implementaron una vez finalizadas las funcionalidades principales del backend. Esto fue en el período comprendido entre mediados de agosto y la tercera semana de septiembre, extendiéndose dos semanas más de lo planificado. Luego, en la última semana de septiembre se llevaron a cabo principalmente correcciones de errores. Respecto de lo planificado, ambas tareas causaron una extensión de dos semanas, a causa de la inexperiencia en la implementación de las pruebas automáticas y lo mucho que se profundizaron. La segunda fase de *testing* fue principalmente manual en el frontend para detectar errores que afecten a la experiencia de usuario y para verificar que se haya integrado todo correctamente con el backend.

- **Despliegue**: para esta etapa se asignaron la tercera y cuarta semana de septiembre, con una estimación inicial de 50 h. El tiempo de ejecución real fue mayor a lo planificado, a causa de la gran cantidad de aspectos técnicos a considerar para hacer un despliegue de una plataforma en producción. En un primer momento, se realizó un despliegue rápido, para que los curadores puedan ingresar a LawCEDIC y aportar *feedback* más preciso. Luego, se incorporaron medidas de seguridad a la máquina virtual para asegurar la disponibilidad y robustez de la plataforma. Una tarea que no se había considerado inicialmente fue la automatización del despliegue ante cambios en el repositorio de GitHub, cuya configuración requirió de varios días. Sin embargo, esto le permitió al equipo ahorrar una gran cantidad de tiempo en el futuro, ya que

para integrar cambios a la rama de desarrollo no es necesario ingresar a la máquina virtual, sino que un script automatizado de Github Actions resuelve esta tarea.

- **Migración de base de datos** : Uno de los cambios más significativos en cuanto a lo planificado fue en este punto. En un comienzo se pensaba realizar un *script* que migre automáticamente los documentos cargados al nuevo sistema. Sin embargo, luego de realizar un análisis de conveniencia en cuanto a tiempo y complejidad se llegó a la conclusión de que la forma más óptima de migración era manual y no automática. Esto se debe a que la base de datos de origen era una relacional generada por Wordpress que contenía toda la información distribuida a lo largo de múltiples tablas mezcladas con metadatos internos de la plataforma, mientras que la base de datos destino era una no relacional con una estructura distinta y datos desnormalizados. Por otro lado, el panel de gestión de Wordpress permitía visualizar los datos más importantes de cada documento mientras que la funcionalidad de carga de documentos de la nueva plataforma ya estaba implementada al momento de la migración. Como última consideración, el volumen de documentos no era significativamente grande, infiriendo que existía la posibilidad de realizarlo manualmente, dividiendo la tarea entre los tres desarrolladores. Este cambio provocó que no tenga sentido comparar el tiempo estimado y el ejecutado ya que las horas estimadas se basaron en una tarea completamente distinta. Es por este motivo que no se incluye esta fase en la figura 20.

- **Capacitación del personal**: las reuniones de capacitación se llevaron a cabo en las semanas planificadas. Para la primera reunión, realizada a inicios de octubre, ya se contaba con la plataforma desplegada, por lo que respetar la planificación de las etapas previas fue crucial para cumplir con las fechas de las capacitaciones. Respecto de la reunión a mediados de noviembre, LawCEDIC contaba con el modelo de IA integrado por completo, por lo que el equipo de desarrollo explicó sus características y funcionalidades prácticas.

- **Integración con modelo de IA**: La integración del modelo de inteligencia artificial representó una fase clave en el desarrollo de la plataforma, dependiendo en gran medida del cumplimiento de los plazos establecidos por la consultora delegada por

Meta para la entrega de la API. Si bien la implementación se completó dentro del período estipulado, su ejecución no fue lineal en términos de carga horaria, ya que requirió múltiples iteraciones, ajustes y cambios tanto en la API proporcionada por la consultora como en la estructura ya creada de la propia plataforma. A lo largo de esta etapa, el equipo de desarrollo mantuvo comunicación constante con los responsables de la consultora para resolver cuestiones técnicas y definir la estrategia óptima de integración. Se llevaron a cabo diversas pruebas tanto de carga como de búsqueda para asegurar que el modelo opere correctamente dentro del flujo de trabajo de la plataforma, garantizando su compatibilidad con el backend y su correcta interacción en el frontend. Gracias a esta colaboración, fue posible lograr una implementación eficiente sin afectar el cronograma general del proyecto. La integración del modelo de IA se completó con éxito y a tiempo, un requerimiento necesario para la presentación de la plataforma en el congreso de San Luis.

- **Documentación de la plataforma**: La documentación principal de la plataforma fue la de la API del backend. Como uno de los desarrolladores se centró a tiempo completo en ella, una vez que finalizó su desarrollo continuó con los *tests* automáticos y luego su documentación. Es por eso que, si bien ya había un boceto de las especificaciones de los *endpoints*, se comenzaron a documentar formalmente en octubre, un mes antes de lo planificado, y llevó una semana menos de lo estimado.

- **Confección de informe y presentación**: El equipo inició la etapa de documentación y presentación del proyecto en los primeros días de diciembre, siguiendo el cronograma establecido. Sin embargo, la confección del informe se vio interrumpida en las últimas dos semanas de diciembre debido a compromisos personales del equipo, como las fiestas y reuniones de fin de año. Este factor, que no había sido contemplado en la planificación inicial, generó una demora en la finalización del documento, extendiendo su cierre hasta mediados de febrero. A pesar de este retraso, la postergación no tuvo un impacto directo en la plataforma en sí, ya que para ese momento LawCEDIC ya se encontraba operativa y en funcionamiento.

  Este aspecto representa un aprendizaje importante para la estimación de tiempos en futuros proyectos, ya que sería apropiado considerar períodos de baja productividad

por eventos externos lo cual permitiría ajustar mejor los cronogramas y evitar retrasos. En la Figura 20 puede observarse gráficamente esta comparación.

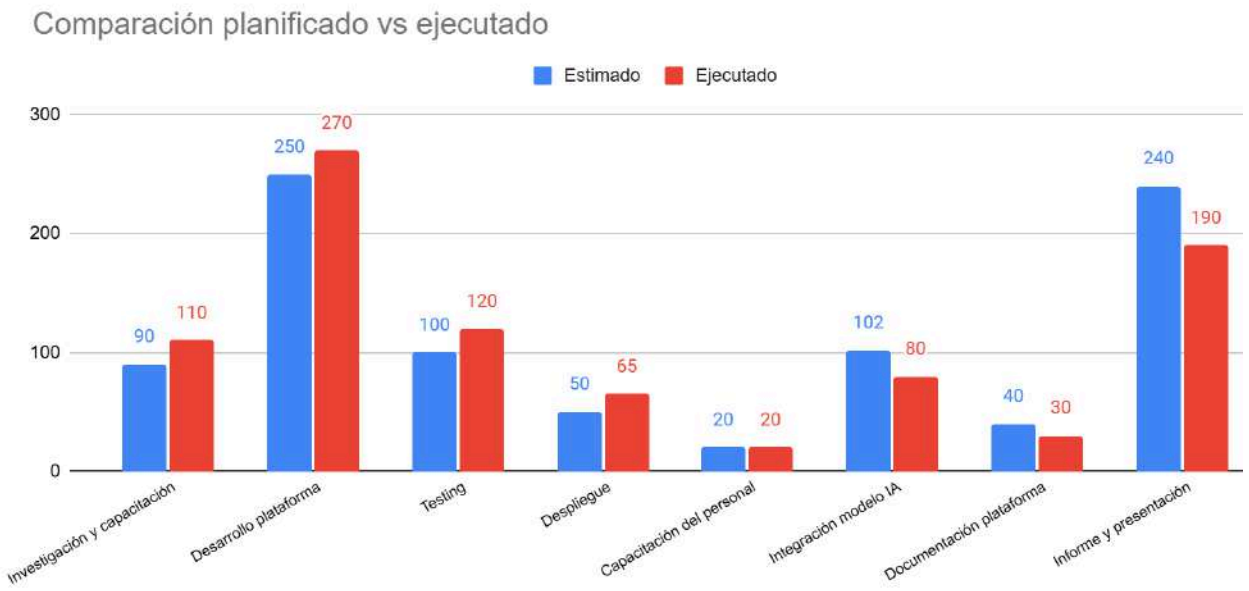

Figura 20: Horas por etapa del proyecto estimadas contra ejecutadas

Al concluir el proyecto, la cantidad total de horas empleadas fue de 1070 horas, lo que en primera instancia representa una reducción con respecto a las 1133 horas inicialmente estimadas. Sin embargo, no es correcto comparar el tiempo de esta manera ya que hubo un cambio significativo en las tareas realizadas para la migración de la base de datos como se mencionó previamente en el detalle de dicha fase. Para tener una mejor comparación no se considerarán las 91 horas estimadas para la migración automática ni las 35 horas empleadas en la manual. De esta manera, el resultado es de 1035 horas empleadas frente a 1042 estimadas, con una desviación de únicamente 7 horas. Esto evidencia que, a pesar de que las horas destinadas a cada tarea en particular variaron respecto de lo estimado, el tiempo total de ejecución del proyecto siguió en línea con la estimación inicial.

7.5. Cambios respecto del FODA inicial

Luego de la ejecución del proyecto se descubrieron y aparecieron algunas características que variaron el FODA inicial.

En primer lugar, se descubrió que la poca experiencia en Inteligencia Artificial no fue una debilidad como se había determinado inicialmente. Esto se debe a que no fue imprescindible poseer conocimientos avanzados en la materia para integrar el modelo en LawCEDIC. La interfaz brindada por la API de Augmented Experiences permitió al equipo abstraerse de la implementación.

Por otro lado, surgieron dos nuevas oportunidades relevantes. La primera fue el convenio entre OCEDIC, Augmented Experiences y Meta, que posibilitó la integración del modelo de inteligencia artificial sin coste alguno para el observatorio. Aunque la tokenización normalmente se monetiza de manera individual por cada consulta al RAG, este acuerdo permitió que se realizara sin cargo para el proyecto.

La segunda oportunidad que apareció, fue el contacto del director del Trabajo Final, Hernán Hinojal, con Ecolan, el centro de datos que proporcionó la solución VPS para el despliegue. Gracias a esta conexión, el equipo tuvo acceso anticipado a la máquina virtual y pudo configurarla antes de la contratación formal del servicio por parte de OCEDIC.

7.6. Presentación en el 4to Encuentro Federal de Cibercrimen y Nuevas Tecnologías

Los días 28 y 29 de noviembre de 2024 se realizó en la Universidad Católica de Cuyo, San Luis, el 4to Encuentro Federal de Cibercrimen y Nuevas Tecnologías organizado por OCEDIC, que contó con la presencia de abogados, analistas, jueces, fiscales, peritos y otros participantes del Poder Judicial. Uno de los paneles de dos horas del congreso fue con motivo de la presentación de la plataforma a nivel técnico y comercial y luego la realización de un juego de roles en la que pueda verse en funcionamiento. El referente funcional de este proyecto, Javier Vellido, insistió en que los desarrolladores de la plataforma asistieran como expositores y estos aceptaron no solo como miembros del equipo, sino en representación de la Universidad Nacional de Mar del Plata.

El segundo día del encuentro tuvo lugar el panel LawCEDIC que comenzó con una introducción técnica al *RAG* y a las técnicas de IA utilizadas encabezada por Tomás Vera y Oscar Cartagena de Augmented Experiences de forma virtual a través de una videollamada proyectada. Luego, los estudiantes presentaron su trabajo, explicando el problema que la plataforma viene a solucionar y sus características principales, siempre considerando que los oyentes provenían, en su mayoría, del ámbito del derecho. Luego de finalizar la presentación y al compartir charlas informales con los asistentes, se confirmó que la búsqueda eficiente de documentos era una problemática común y generalizada, por lo que LawCEDIC generó una gran expectativa en ellos.

Para los estudiantes, haber sido parte de un evento de esta relevancia representando a su universidad fue una experiencia enriquecedora. Este espacio representó una gran oportunidad para validar el producto con usuarios potenciales y recibir retroalimentación directa sobre su utilidad y posibles mejoras. Escuchar las opiniones, sugerencias y críticas de quienes podrían utilizar LawCEDIC en su día a día permitió identificar aspectos que podrían mejorarse y confirmar cuáles son las funcionalidades más valoradas. También fue una instancia clave para poner a prueba su oratoria y habilidades de comunicación, permitiéndoles desarrollar la capacidad de transmitir ideas de manera clara y adaptada a diferentes audiencias. Sin duda, esta experiencia dejó un aprendizaje valioso en el crecimiento profesional y académico del equipo.

Figura 21: Fotos tomadas en el 4to Encuentro Federal de Cibercrimen y Nuevas Tecnologías en la Universidad Católica de Cuyo, San Luis.

8. Conclusiones y trabajos futuros

El desarrollo de LawCEDIC ha permitido la creación de una plataforma web innovadora para la gestión de documentos judiciales vinculados al cibercrimen y la evidencia digital. Su objetivo principal ha sido mejorar el acceso y la organización de la información para profesionales del derecho. De esta manera, se optimizó la carga, búsqueda y clasificación de documentos mediante inteligencia artificial, permitiendo consultas más eficientes mediante lenguaje natural y reduciendo el tiempo requerido para gestionar grandes volúmenes de documentación.

La planificación inicial fue un factor crítico considerando que, más allá del plazo de entrega de este informe, gran parte de la plataforma debía estar funcional para su presentación en el 4to Encuentro Federal de Cibercrimen y Nuevas Tecnologías. Aunque la distribución de tiempos no se ajustó estrictamente a lo previsto, la flexibilidad del equipo permitió adaptarse a los cambios, de forma que se finalizó el trabajo en tiempo y forma.

La presentación de LawCEDIC en el congreso representó un reto importante en términos de exposición pública y permitió una mejor validación del trabajo frente a profesionales del derecho, potenciales usuarios del sistema. Asimismo, el trabajo interdisciplinario requirió explicar conceptos técnicos de manera clara y accesible, asegurando que todas las partes involucradas comprendieran el funcionamiento y los alcances de la herramienta. En consecuencia, fue fundamental desarrollar competencias en oratoria y comunicación eficaz.

La capitalización de los desaciertos en forma de aprendizaje es fundamental, sobre todo en la estimación y gestión del tiempo en proyectos de esta magnitud. Uno de los errores fue haber asumido que las horas de trabajo de todas las semanas se iban a mantener constantes, sin contemplar otras responsabilidades o semanas de baja productividad. El otro fallo fundamental fue no haber documentado debidamente todas las tareas realizadas y su duración, lo que provocó una pérdida de la trazabilidad del tiempo empleado que volvió difícil la comparación con el estimado para finalmente evaluar el resultado del trabajo. Sin embargo, la gestión de proyectos se aprende con la práctica, por lo que esta experiencia resultó enriquecedora para los estudiantes en su formación como futuros profesionales.

El impacto de LawCEDIC ya es tangible en cuanto a la automatización del proceso de carga de documentos a la plataforma. La búsqueda mediante lenguaje natural, si bien ya está implementada y funcional, aún no fue lanzada para su utilización por los usuarios finales. Sin embargo, aún existen oportunidades de mejora y expansión, como la incorporación de nuevas funcionalidades para mejorar la experiencia del usuario. Además, la posibilidad de comercializar el sistema abre un nuevo horizonte para su sostenibilidad y crecimiento, permitiendo incorporar a los desarrolladores de forma oficial al equipo de OCEDIC para su mantenimiento.

En conclusión, el desarrollo de LawCEDIC ha sido un desafío que demandó esfuerzo y compromiso para abordar una problemática real, pero cuyos resultados reflejan un impacto positivo y un camino prometedor para futuras mejoras e innovaciones.

8.1. Trabajos a futuro

El siguiente gran paso de LawCEDIC es su apertura al público general ya que, si bien ya está en funcionamiento, todavía está en una fase de beta cerrada, en la que pocas personas tienen acceso. Además, se recolectaron una serie de ideas o deseos de varios de los integrantes del equipo que se planean implementar en un futuro.

Una de las ideas es que el usuario no solo disponga de carpetas de favoritos de uso personal, sino que pueda compartir un espacio de trabajo con otros usuarios. Es decir, que varios usuarios de una misma organización puedan compartir un mismo directorio de documentos para agrupar casos de interés y facilitar la colaboración.

Otra de las sugerencias es la generación de citas de los documentos en distintos formatos (normas APA, *The Bluebook: A Uniform System of Citation*, pautas particulares de jurisprudencia, entre otros) con toda la información requerida y su exportación automática.

Una característica menor pero interesante para garantizar la mejor experiencia de usuario es la capacidad de personalizar la plataforma. Esto abarca desde poder elegir la paleta de colores para que sea más agradable, hasta poder guardar preferencias de filtros frecuentes como etiquetas o tipos de delitos, para no tener que volver a seleccionarlos cada vez que se ingrese.

Por último, un gran proyecto que se desprende de LawCEDIC y que se cree muy innovador es un servicio de anonimización automática de documentos tanto para su carga a la plataforma como para cualquier usuario que quiera censurar la información sensible de un documento judicial. Con inteligencia artificial se podrían detectar de forma automática todos los datos que se quieran resguardar, incluyendo nombres, domicilio y datos identificatorios de las partes involucradas.

9. Glosario

- **API *(Application Programming Interface):*** Es un conjunto de reglas y definiciones que permiten la comunicación entre diferentes aplicaciones o sistemas. Facilita el intercambio de datos y funcionalidades a través de protocolos como HTTP en servicios web.

- **API-Key:** Es una clave única que se utiliza para autenticar y autorizar el acceso a una API (Application Programming Interface). Funciona como un identificador y mecanismo de seguridad para controlar quién puede usar la API y qué acciones puede realizar. Se utilizan para prevenir el uso no autorizado y limitar el número de solicitudes según las políticas del servicio.

- ***Cookie***: Es un pequeño archivo de datos que un sitio web almacena en el navegador del usuario para guardar información sobre su sesión, preferencias o actividad. Se usan para autenticación, seguimiento de usuarios y personalización de contenido.

- ***Commit***: Es una acción en los sistemas de control de versiones, como Git, que guarda cambios en el historial del repositorio. Cada commit crea un punto de referencia con un mensaje descriptivo, permitiendo rastrear modificaciones y revertir cambios si es necesario.

- **DOM (Document Object Model):** Es una representación estructurada de un documento HTML o XML en forma de un árbol de nodos. Cada elemento, atributo o contenido del documento es un nodo en esta estructura, lo que permite a los lenguajes de programación como JavaScript interactuar y manipular dinámicamente el contenido, la estructura y el estilo de la página web.

- ***Endpoint***: Es una URL específica dentro de una API donde se pueden enviar solicitudes para interactuar con un recurso. Define una ruta accesible por clientes para realizar operaciones como obtener, enviar, actualizar o eliminar datos mediante métodos HTTP (GET, POST, PUT, DELETE).

- ***Hash***: Es una función que toma una entrada y devuelve un valor fijo de longitud corta, generalmente representado como una cadena alfanumérica. Se utiliza para

verificar la integridad de los datos, crear identificadores únicos o almacenar contraseñas de forma segura, ya que es prácticamente imposible revertir el valor hash a su forma original.

- ***Headers***: Son metadatos incluidos en las solicitudes y respuestas HTTP que proporcionan información adicional, como el tipo de contenido, la autenticación, el origen de la petición y el control de caché. Ayudan a definir el comportamiento y la seguridad de la comunicación entre cliente y servidor.

- **Jails**: Configuraciones específicas que monitorizan registros del sistema para detectar patrones de actividad maliciosa, como intentos fallidos de inicio de sesión. Cuando se identifica una actividad sospechosa, la *jail* correspondiente bloquea temporalmente la dirección IP del atacante para prevenir nuevos intentos.

- **LLM (Large Language Model):** Son modelos de lenguaje de gran escala entrenados con enormes cantidades de datos textuales para comprender y generar lenguaje humano. Utilizan redes neuronales profundas para predecir y generar texto coherente en función del contexto proporcionado. Los LLM pueden realizar tareas como redacción, traducción, resumen de textos y generación de código, entre muchas otras, con un nivel de sofisticación que imita el lenguaje natural.

- ***Logging***: Es el proceso de registrar eventos, errores o información relevante sobre la ejecución de una aplicación. Se usa para monitoreo, depuración y análisis de rendimiento, y puede almacenarse en archivos, bases de datos o sistemas de gestión de logs.

- ***Middleware***: Es un software que actúa como intermediario entre diferentes aplicaciones o servicios, facilitando la comunicación, la gestión de datos y la integración entre sistemas. En el desarrollo web, se usa comúnmente en frameworks como Express.js para manejar peticiones HTTP antes de que lleguen a las rutas específicas.

- **Proxy**: Intermediario que actúa entre el cliente y el servidor de destino, reenviando las solicitudes y respuestas. Se utiliza para mejorar la seguridad, anonimizar la

navegación, equilibrar la carga de tráfico y almacenar en caché contenido para mejorar el rendimiento.

- **RAG (Retrieval-Augmented Generation):** Es una técnica de inteligencia artificial que combina la generación de texto mediante modelos de lenguaje (LLM) con recuperación de información de fuentes externas. En RAG, el modelo primero recupera datos relevantes de una base de conocimientos o documentos y luego los utiliza para generar respuestas más precisas y contextuales.

- **RESTful**: Hace referencia a sistemas o servicios web que siguen los principios de REST (*Representational State Transfer*). Un servicio RESTful utiliza métodos HTTP (GET, POST, PUT, DELETE) para la manipulación de recursos y sigue convenciones estandarizadas para la estructuración de URLs y respuestas en formatos como JSON o XML.

- **SPA (*Single Page Application*)**: Es un tipo de aplicación web que carga una única página HTML y actualiza dinámicamente su contenido mediante JavaScript, sin necesidad de recargar la página completa. Esto mejora la velocidad y la experiencia del usuario.

- ***Stateless***: Se refiere a un sistema o servicio que no guarda el estado de las interacciones previas entre el cliente y el servidor. Cada solicitud es independiente y debe contener toda la información necesaria para ser procesada correctamente.

- ***Stored procedures***: Son conjuntos de instrucciones SQL precompiladas que se almacenan y ejecutan en una base de datos. Se usan para mejorar el rendimiento, reducir la redundancia en consultas repetitivas y mejorar la seguridad al evitar la inyección de SQL. En el caso actual, como la base de datos es no relacional se utilizan las llamadas JavaScript functions que se almacenan en el servidor y se ejecutan sobre la base de datos.

- **TDD (*Test Driven Development*)**: Es una metodología de desarrollo de software en la que los tests se escriben antes de escribir el código de la funcionalidad que deben

verificar. El ciclo TDD sigue tres pasos: escribir un test que falle, escribir el código necesario para pasar el test y refactorizar el código si es necesario.

- **Tokenización del texto**: Es el proceso de dividir un texto en unidades más pequeñas llamadas *tokens*, que pueden ser palabras, frases o caracteres. Se usa en procesamiento de lenguaje natural (NLP) para estructurar texto de manera que pueda ser analizado y procesado por algoritmos.

- ***Triggers***: Son procedimientos almacenados en la base de datos que se ejecutan automáticamente cuando ocurre un evento específico, como la inserción, actualización o eliminación de un registro en una tabla o de un documento, en caso de MongoDB. Se utilizan para garantizar la integridad de los datos y automatizar procesos dentro de la base de datos.

- **URL (*Uniform Resource Locator*)**: Es la dirección única que identifica un recurso en la web. Se compone de varios elementos, como el protocolo (HTTP/HTTPS), el dominio, la ruta y, opcionalmente, parámetros y fragmentos.

- **VPS (*Virtual Private Server)***: Es un servidor virtualizado dentro de un servidor físico que actúa como un servidor dedicado, pero con recursos compartidos. Se usa para alojar sitios web, aplicaciones o bases de datos, ofreciendo mayor control y flexibilidad que un hosting compartido, pero sin el costo de un servidor dedicado.

10. Anexos

10.1. Documentación API Rest

Para poder visualizar la documentación de la API es necesario descargar el archivo “Swagger UI.html” y la carpeta “Swagger UI_files” presentes en la carpeta “Documentación API Rest”, y ubicarlos en el mismo directorio. Luego, abrir el archivo “Swagger UI.html”.

10.2. Guía de administración de contenedores Docker

En el archivo “Guía de administración de contenedores Docker - LawCEDIC” se incluyen los comandos necesarios para realizar tareas de mantenimiento y supervisión de los contenedores, asegurando el correcto funcionamiento del sistema desplegado.

10.3. *Scripts backup* de la base de datos

En la carpeta “Scripts backup” se encuentran los archivos “script-backup.sh” y “script-restaurar-backup.sh”. El primero realiza las copias de seguridad de la base de datos MongoDB y de los archivos de logs del sistema, mientras que el segundo permite restaurar las copias de seguridad generadas.

10.4. *Scripts* despliegue automatizado ante cambios

En la carpeta “Scripts despliegue” se encuentran “script-githubActions-frontend.yml”, “script-githubActions-backend.yml” y “workflow-principal.yml”. Los primeros dos se ejecutan ante cambios en las ramas production de sus respectivos repositorios y al hacerlo activan el tercero, encargado de todo el proceso de despliegue.

10.5. *Dockerfiles*

En la carpeta “Dockerfiles” se encuentra el “docker-compose.yml”, el cual lleva a cabo la orquestación de los contenedores en la máquina virtual. Además, el “dockerfile-frontend.txt” y el “dockerfile-backend.txt", que definen las configuraciones de los contenedores para cada servicio.

10.6. Especificación de comunicación con servicios de IA

En el archivo “Especificación de comunicación con servicios de IA” se encuentra la interfaz utilizada entre el backend de la plataforma y los servicios de extracción de datos de documentos, edición de atributos y búsqueda de documentos proporcionados por Augmented Experiences.

11. Bibliografía

1. Web oficial GitHub.
   https://github.com/
2. Documentación de React.
   https://react.dev/reference/react
3. Documentación de librería React Dropzone.
   https://react-dropzone.js.org/
4. Documentación librería React Select.
   https://react-select.com/home
5. Documentación librería React Slick.
   https://react-slick.neostack.com/
6. Documentación librería React Chartjs2.
   https://react-chartjs-2.js.org/
7. Documentación librería tsParticles.
   https://particles.js.org/docs/index.html
8. Documentación librería React Pro Sidebar.
   https://www.npmjs.com/package/react-pro-sidebar
9. Documentación librería icomoon-react.
   https://www.npmjs.com/package/icomoon-react
10. Tutorial despliegue automático con Github Actions:
    https://medium.com/@avash700/ci-cd-made-easy-github-actions-docker-compose-and-watchtower-60a698d24f27
11. Documentación Docker Compose:
    https://docs.docker.com/compose/
12. Configuración del firewall UFW:

https://www.digitalocean.com/community/tutorials/how-to-set-up-a-firewall-with-ufw-on-ubuntu

13. Gestión de autenticación frontend:

    https://dev.to/miracool/how-to-manage-user-authentication-with-react-js-3ic5

14. Documentación Nginx:

    https://nginx.org/en/

15. Configuración MongoDB en Docker Compose:

    https://www.mongodb.com/resources/products/compatibilities/docker

16. Documentación Node.js:
    https://nodejs.org/docs/latest/api/

17. Documentación Express.js
    https://expressjs.com/en/5x/api.html

18. Documentación mongoose:

    https://mongoosejs.com/docs/

19. Documentación JSON Web Token:

    https://jwt.io/introduction

20. Documentación Passport:

    https://www.passportjs.org/docs/

21. Fail2ban:

    https://github.com/fail2ban/fail2ban

22. Documentación Swagger:

    https://swagger.io/docs/

23. Documentación Multer

https://www.npmjs.com/package/multer

24. Documentación Chai:

https://www.chaijs.com/

25. Documentación Mocha:

https://mochajs.org/api/

26. Alternativas de despliegue - IaaS vs PaaS vs SaaS

https://www.redhat.com/es/topics/cloud-computing/iaas-vs-paas-vs-saas

27. Documentación Vite:

https://vite.dev/guide/build

28. Documentación TailwindCss:

https://v2.tailwindcss.com/docs

29. Interceptores en Axios:

https://lightrains.com/blogs/axios-intercepetors-react/

30. Documentación Axios:

https://axios-http.com/docs/intro

31. Instalación de certificados SSL:

https://www.digicert.com/kb/csr-ssl-installation/nginx-openssl.htm

32. Proxy y reverse proxy:

https://www.cloudflare.com/learning/cdn/glossary/reverse-proxy/