Pontificia Universidad Javeriana Cali
Facultad de Ingeniería.
Ingeniería de Sistemas y Computación.
Proyecto de Grado.

Diseño de una arquitectura distribuida para la sincronización de procesos judiciales

Brian Alexis Mosquera Tunubalá

Director: Royer David Estrada Esponda

30 de Octubre de 2024

Santiago de Cali, 30 de Octubre de 2024.

Señores
Pontificia Universidad Javeriana Cali.
Ph.D. Luisa Rincón
Directora Maestría en Ingeniería de Software.
Cali.

Cordial Saludo.

Por medio de la presente hago constar que en mi calidad de director de trabajo de grado he revisado el proyecto titulado "Diseño de una arquitectura distribuida para la sincronización de procesos judiciales" realizado por el estudiante de Ingeniería de Sistemas y Computación Brian Alexis Mosquera Tunubalá (cod: 8955561), el cual se encuentra terminado y considero que cumple con los requisitos para ser sustentado.
Atentamente,

Royer David Estrada Esponda

Santiago de Cali, 30 de Octubre de 2024.

Señores
Pontificia Universidad Javeriana Cali.
Ph.D. Luisa Rincón
Directora Maestría en Ingeniería de Software.
Cali.

Cordial Saludo.

Me permito presentar a su consideración el proyecto de grado titulado "Diseño de una arquitectura distribuida para la sincronización de procesos judiciales" con el fin de cumplir con los requisitos exigidos por la Universidad y para que sea sometido a revisión del jurado y cumpla su aprobación, para conseguir posteriormente el título de Ingeniero de Sistemas y Computación.

Atentamente,

Brian Alexis Mosquera Tunubalá
Código: 8955561

Santiago de Cali, 30 de Octubre de 2024.

Señores
Pontificia Universidad Javeriana Cali.
Ph.D. Luisa Rincón
Directora Maestría en Ingeniería de Software.
Cali.

Cordial Saludo.

Por medio de la presente, Redjudicial expresa su conformidad y acepta el manejo, tratamiento, y publicación de los datos utilizados en el marco del proyecto titulado "Diseño de una arquitectura distribuida para la sincronización de procesos judiciales" de conformidad con las regulaciones vigentes sobre protección de datos y privacidad.
Los datos utilizados en este proyecto están relacionados con nuestros productos, servicios y tecnologías usadas, los cuales han sido recopilados, procesados y analizados de acuerdo a las normativas aplicables.
Asimismo, garantizamos que el uso y la publicación de estos datos tienen como propósito exclusivo la mejora del sistema Redjudicial, enfocado en la optimización de los procesos judiciales y la automatización de la sincronización de información relevante. En ningún caso se divulgarán datos personales, sensibles o información que comprometa la privacidad de nuestros clientes o la confidencialidad de los casos judiciales.
Atentamente,

Rubén Adolfo Pino
Representante legal Redjudicial

0.1. Resumen

Este proyecto de grado propone el diseño de una arquitectura de software distribuido para la empresa Redjudicial con el objetivo de automatizar la sincronización de procesos judiciales y poder así mejorar el rendimiento, la fiabilidad y la tolerancia a fallos del sistema.

Este proyecto se desarrolló usando la metodología Scrum junto con el enfoque As-is-to-be en el cual se identifica el estado actual del sistema y los procesos existentes (As-Is) para diseñar a partir de este un estado futuro (To-be) en el cual se apliquen las oportunidades de mejora evidenciadas. La combinación de un enfoque ágil como Scrum y el enfoque As-Is-To-Be permitió una gestión dinámica y estructurada del proyecto, asegurando la entrega de un producto de alta calidad que cumpla con los atributos de calidad planteados por el cliente.

Este proyecto establece una base tecnológica para automatizar el proceso de sincronización de procesos judiciales de manera eficiente, confiable y escalable. La arquitectura propuesta cumple con los requisitos técnicos y estratégicos, asegurando la reducción de procesos manuales en Redjudicial.

Palabras Clave: Extracción Web , Procesamiento distribuido, Arquitectura de software, Concurrencia, Procesamiento paralelo, Tolerancia a fallos

0.2. Abstract

Keywords: Web scraping, Distributed processing, Software architecture, Concurrency, Parallelism, fault tolerance

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Definición del problema de investigación

La arquitectura de software de una organización es el conjunto de decisiones técnicas que determinan la estructura, el comportamiento, interacción y la evolución de un sistema software. Tiene como objetivo optimizar los recursos requeridos para poder desarrollar y mantener un sistema (Martin, 2017).

Este concepto surge en la década de 1950 como respuesta a la necesidad de desarrollar software complejo. (Taylor, 2021), lo cual también implicó una evolución en cuanto a métodos y herramientas para abordar los requerimientos funcionales y de calidad que desafiaban a los sistemas existentes.

La arquitectura de software implica para las organizaciones una definición y priorización de necesidades, con las cuales el negocio establece qué aspectos son más importantes para asegurar en sus sistemas y enfocar su arquitectura en estos. Aunque una adecuada planificación y diseño pueden ofrecer grandes beneficios para una organización, es común que para empresas de pequeño y mediano tamaño este proceso sea más complejo por las implicaciones que pueden tener en cuestión de costos, cultura organizacional y sostenibilidad en el tiempo.

Redjudicial es una empresa caleña de servicios tecnológicos que desde el 2006 ofrece una plataforma para abogados y despachos jurídicos que les permite gestionar los procesos judiciales y automatizar la actualización de estos, de tal manera que sus clientes no tengan que consultar a diario toda la información disponible en los juzgados y en los sistemas de la rama judicial.

Para esto, la empresa utiliza el aplicativo de la rama judicial colombiana que permite consultar en línea la información de los procesos a través de un número de radicado. Al principio, esta tarea se hacía de forma manual, descargando la información e insertándola en su base de datos a través de archivos de Excel. Posteriormente, implementaron Web Scraping y finalmente desarrollaron un aplicativo en netcore que se encarga de descargarlos usando paralelismo, pero aún tienen una gran parte de los procesos que se consultan de forma manual, ya que diariamente se deben revisar más de 16.000 procesos.

Todo este proceso de desarrollo se ha realizado de manera aislada, pensando en resolver una necesidad específica sin tener en cuenta todos los atributos de calidad que el sistema requiere o podría requerir, lo que ha ocasionado que constantemente se hagan cambios en el aplicativo. Algunos de ellos han implicado el desarrollo de un nuevo sistema en una tecnología diferente, lo que también genera costos extras para la compañía.

Debido a la gran cantidad de procesos y a las limitaciones de *Rama judicial*, este proceso no se ha podido automatizar completamente, ya que para aumentar la capacidad del sistema deben aumentar los recursos de la máquina, lo cual permite consultar más procesos simultáneamente, pero

después de cierto tiempo con una alta demanda, la disponibilidad de *Rama judicial* se degrada, lo cual lleva a que el sistema falle y tenga que volver a iniciar, retrasando así la actualización de procesos.
Estos retrasos han causado que algunos de sus clientes se molesten, ya que no reciben oportunamente la notificación de sus procesos y que la empresa tenga que contratar a más personas para realizar la parte faltante del proceso, generando así costos demasiado altos.

1.1.1. Planteamiento del problema

Lo anterior genera las siguientes preguntas: ¿Cómo hacer más eficiente la consulta de información de *Rama judicial*? ¿Cómo puede el sistema de Redjudicial gestionar los fallos del sistema *Rama judicial*? ¿Cómo lograr que el extractor pueda seguir evolucionando sin que implique reprocesos para la organización?

1.1.2. Sistematización

1. En el proceso de sincronización de procesos judiciales ¿cuáles son los atributos de calidad más importantes para Redjudicial?

2. ¿Cuáles son los principales desafíos técnicos y operativos que enfrenta Redjudicial al evolucionar su sistema de automatización de procesos?

3. ¿Cuáles son los factores que ocasionan las fallas en la integración entre Ramajudicial y el extractor?

1.2. Objetivos

1.2.1. Objetivo General

Implementar una arquitectura de software eficiente y escalable para Redjudicial que permita la consulta eficiente de información de la Rama Judicial, gestione de manera efectiva los fallos del sistema de la Rama Judicial y facilite la evolución continua del extractor sin generar reprocesos en la organización.

1.2.2. Objetivos específicos

1.2.2.1. Realizar un análisis exhaustivo de los procesos actuales de obtención de datos de la Rama Judicial, identificando las ineficiencias y los puntos críticos que requieren mejoras.

1.2.2.2. Diseñar una arquitectura de software que optimice la consulta de información de la Rama Judicial, priorizando la eficiencia y la escalabilidad, y minimizando la necesidad de aumentar los recursos de hardware.

1.2.2.3. Desarrollar un mecanismo robusto para la gestión de fallos del sistema de la Rama Judicial, que permita una recuperación rápida y efectiva en caso de degradación de la disponibilidad.

1.2.2.4. Evaluar y validar la arquitectura propuesta a través de pruebas exhaustivas y mediciones de rendimiento, asegurando que cumpla con los criterios de calidad y eficiencia definidos.

1.3. Justificación del trabajo de grado

El uso adecuado de las tecnologías de la información permite a las organizaciones operar de manera eficiente, lo cual a su vez permite que estas puedan crecer sin que su productividad y calidad de servicio se vean afectadas por la tecnología que usan.

Para Redjudicial, la implementación de una arquitectura distribuida para la sincronización de sus procesos judiciales es una gran oportunidad de mejora, ya que puede aprovechar mejor los recursos computacionales con los que cuenta y reducir los costos de operación, teniendo en cuenta que con esta arquitectura se pudo automatizar un proceso que, por ser manual, implicaba gastos en nómina o aumento en la operatividad (para el personal actual) además de incrementar su capacidad de procesamiento.
Este aumento de la capacidad de procesamiento fue un gran hito para la compañía, ya que asegura su operación en el tiempo, pues podrá seguir creciendo con la tranquilidad de que su base tecnológica podrá evolucionar a la par del negocio.

No haber realizado este proyecto tendría serias implicaciones para Redjudicial ya que continuaría con un proceso manual que incrementaría sus costos y reduciría su productividad, sería arriesgado ya que está abierto a errores humanos y estos a su vez se pueden convertir en una mala experiencia para sus clientes y en ciertos casos hasta en problemas legales, además de que la empresa no estaría preparada tecnológicamente para afrontar los desafíos que surjan con el crecimiento del negocio.
Desde el punto de vista teórico, existen estudios que demuestran que el uso de concurrencia y paralelismo proporciona un aumento significativo en el rendimiento, permitiendo que las tareas se completen más rápido en grandes cantidades de información (Tanenbaum and van Steen, 2023), esta premisa permitió considerar que el procesamiento distribuido puede satisfacer las necesidades

de mejora de Redjudicial.
Desde el punto de vista del aporte teórico, y con base en la revisión inicial de la literatura, el proyecto era justificable debido a que se sabe que el procesamiento distribuido tiene impactos positivos en el rendimiento de las aplicaciones al dividir el trabajo entre múltiples nodos y esto puede conducir a un tiempo de ejecución más rápido, ya que las tareas se pueden completar de manera simultánea (Tanenbaum and van Steen, 2023).
Finalmente, en términos prácticos, se puede señalar que el proyecto se justifica debido a la vigencia que tiene el problema seleccionado a nivel general, ya que la automatización de procesos, que según algunas publicaciones (Deloitte, 2023) es una necesidad urgente para las organizaciones.

1.4. Delimitaciones y Alcance

Para el presente proyecto se documentó el estado actual del proceso de sincronización de procesos judiciales en el cual se identificaron cuáles son las falencias y las deficiencias técnicas que las causan o que no han permitido que se aborden los requerimientos.
Se diseñó la arquitectura del sistema con base en las necesidades u oportunidades de mejora identificadas en el análisis previo, este diseño estuvo sujeto a la aprobación de la compañía, lo cual también quedó documentado en las decisiones técnicas. En este diseño se tuvieron en cuenta las posibles fallas externas y se definieron estrategias para abordarlas; se determinaron las metodologías, tecnologías y herramientas a usar para desarrollo, monitoreo y despliegue.
Esta arquitectura se puso a prueba con componentes básicos que dieron respuesta a los requerimientos de calidad y que permitieron validar que el diseño cumplía con los requisitos de calidad del sistema.
Dentro de la implementación de este proyecto no se incluyeron componentes DevOps, estos se mencionaron, se consideraron dentro del diseño, pero no se implementaron ya que será la empresa la encargada de hacerlo.
Todas estas prácticas, metodologías y herramientas a usar quedaron documentadas y fueron socializadas a los interesados del proyecto.
Dentro de este proyecto solo se consideran los casos de uso de sincronización de procesos judiciales y notificación de nuevas actuaciones, cualquier otro caso de uso solo será referenciado por el impacto que pueda tener en este proyecto o que este proyecto pueda tener sobre ellos.

Es importante tener en cuenta que por políticas de confidencialidad de Redjudicial, dentro de los entregables de este proyecto no se incluyó el código fuente de los componentes que se desarrollaron, ya que estos pertenecen exclusivamente a Redjudicial, aunque se incluyeron métricas y diagramas de alto nivel. Para todos los artefactos generados se definió el nivel de privacidad de ellos y este proceso quedó documentado.

1.4.1. Resultados esperados

1. Documentación Detallada de Procesos Actuales: Una documentación completa de los procesos actuales de obtención de datos de la Rama Judicial, incluyendo flujos de trabajo, sistemas involucrados y responsabilidades.

2. Diseño de Arquitectura de Software: Una arquitectura de software diseñada y documentada que optimiza la consulta de información de la Rama Judicial, priorizando la eficiencia y la escalabilidad, y minimizando la necesidad de aumentar los recursos de hardware.

3. Implementación de Mecanismo de Gestión de Fallos: Un mecanismo robusto implementado para la gestión de fallos del sistema de la Rama Judicial, permitiendo una recuperación rápida y efectiva en caso de degradación de la disponibilidad.

4. Proceso de Desarrollo Continuo Establecido: Un proceso establecido para el desarrollo y evolución continua del extractor de datos que garantice que los cambios y mejoras se puedan implementar sin interrumpir la operación normal de la organización.

5. Validación y Pruebas: Resultados de pruebas exhaustivas y mediciones de rendimiento que validan que la arquitectura cumple con los criterios de calidad y eficiencia definidos.

Capítulo 2

Marco de referencia

2.1. Marco contextual

Redjudicial es una empresa caleña de servicios tecnológicos que desde el 2006 ofrece una plataforma para abogados y despachos jurídicos que les permite gestionar los procesos judiciales y automatizar la actualización de estos, de tal manera que sus clientes no tengan que consultar a diario toda la información disponible en los juzgados y en los sistemas de la rama judicial.
Redjudicial funciona a través de dos empresas REDJUDICIAL SAS y SISTEMA JUDICIAL WEB SAS, las cuales prestan un servicio integral para despachos de abogados y departamentos jurídicos de empresas y entidades públicas, siendo Redjudicial pionera a nivel nacional en el uso de la tecnología de Internet para la vigilancia y control de procesos judiciales, minimizando tiempo y costos a sus más de 1.200 usuarios.
Sistema judicial nace como una herramienta virtual para complementar los servicios de seguimiento de los procesos con fotos de las providencias, estudios y depuración de listas de expedientes, administración y actualización de la información y el manejo de la misma en la nube, en nuestro Software de gestión de procesos, con manejo de clientes, personal, gestión documental, ruta procesal, agenda y auditoría, historial de movimientos y mucho más.

2.1.1. Antecedentes y contexto de Redjudicial:

A través de los servicios prestados, Redjudicial busca mejorar la eficiencia y la transparencia en el ámbito judicial a través de soluciones tecnológicas innovadoras, adaptadas a las necesidades específicas del sector legal en Colombia. Su objetivo es contribuir a la modernización y la transformación digital de sus clientes, ofreciendo herramientas que faciliten el trabajo de abogados, jueces y otros profesionales del derecho.
En este proceso de transformación, Redjudicial ha desarrollado un extractor capaz de consultar información directamente desde el aplicativo Ramajudicial y registrarla en sus sistemas para mantener actualizados los procesos judiciales registrados.

2.2. Bases Teóricas

A través de la revisión sistemática de la literatura se lograron identificar proyectos desarrollados para solucionar problemas parecidos al planteado en este proyecto de trabajo de grado, por lo cual se presentan conceptos necesarios para el entendimiento de este proyecto y que servirán como referentes teóricos para el desarrollo del mismo.

2.2.1. Arquitectura de Software:

A través del tiempo se han elaborado diferentes conceptos o definiciones para la arquitectura de software, las cuales permiten abordar el diseño y la planificación de sistemas desde diferentes enfoques.

La arquitectura de software se puede abordar desde una perspectiva académica como una disciplina técnica y de gestión, considerando aspectos como la estructura, los componentes y su interacción, sus propiedades y cómo interactúan para cumplir los requisitos funcionales y de calidad del sistema. (Bass et al., 2012)

Desde un enfoque de diseño, la arquitectura de software es el conjunto de decisiones sobre la organización de un sistema de bajo y alto nivel, estableciendo principios para la creación de sistemas fáciles de entender, mantener y adaptar a las nuevas necesidades del negocio. (Martin, 2017)

Desde un enfoque mucho más práctico se puede definir a través de las técnicas, patrones y tácticas que se usan en el diseño de arquitecturas, resaltando en este proceso la importancia de la comunicación efectiva y la documentación clara en el proceso de diseño y evolución del sistema. (Fowler et al., 2002)

También se puede ver desde una perspectiva multidimensional, como la estructura fundamental del sistema y la forma en la que los componentes interactúan entre ellos, las decisiones arquitectónicas en aspectos como escalabilidad, rendimiento y seguridad (Richards and Ford, 2020)

De todo esto se puede decir que la arquitectura de software puede tener diferentes enfoques que deben ser considerados y cubiertos para la creación y evolución de sistemas de información de manera óptima, siguiendo principios que permitan la definición de los componentes que la conforman y la interacción entre estos, así como los objetivos con los que se guían las decisiones técnicas a través de las cuales se define el diseño y la evolución del sistema.

2.2.2. Principios de diseño:

Los principios de diseño son directrices fundamentales para la creación de la estructura y organización de un sistema de software. Estos principios son esenciales en la toma de decisiones para garantizar que la arquitectura sea robusta, flexible y cumpla con los requisitos de calidad del sistema. A continuación, se describen algunos de los principales principios de diseño de arquitectura de software.

- **Principio de Separación de Responsabilidades (SRP):** Este principio establece que cada componente o artefacto de un sistema debe tener una única responsabilidad, por lo cual, una razón para cambiar. Esto mejora la cohesión y facilita la comprensión y mantenimiento del código.
- **Principio de Abierto/Cerrado (OCP):** Según este principio, las entidades del sistema deben estar abiertas para su extensión pero cerradas para su modificación. Esto se logra mediante el uso de interfaces y patrones de diseño que permiten agregar nuevas funcionalidades sin alterar el código existente.

- **Principio de Sustitución de Liskov (LSP):** Este principio enfatiza que los objetos de un programa deben poder ser reemplazados por instancias de sus subtipos sin alterar la corrección del programa. Esto significa diseñar componentes que sean intercambiables sin afectar el comportamiento general del sistema. Por ejemplo, al diseñar interfaces y clases base, se asegura que las implementaciones puedan ser sustituidas por subclases sin romper la funcionalidad.
- **Principio de Segregación de Interfaces (ISP):** Este principio propone que las interfaces de un componente deben ser específicas para los clientes que las utilizan, evitando interfaces demasiado generales que obliguen a los clientes a depender de funcionalidades que no necesitan.
- **Principio de Inversión de Dependencias (DIP):** Este principio sugiere que las dependencias deben ser hacia abstracciones y no hacia implementaciones concretas, lo que permite una mayor flexibilidad y facilidad de cambio.
- **Testeabilidad:** La arquitectura debe facilitar la escritura de pruebas unitarias y de integración, permitiendo validar el funcionamiento del sistema de manera efectiva.
- **Independencia de la lógica de negocio:** Las reglas de negocio deben ser independientes de las implementaciones técnicas, lo que facilita su comprensión y modificación.

2.2.3. Atributos de calidad

Los atributos de calidad son características fundamentales que describen el grado en que un sistema o producto de software cumple con los requisitos establecidos para satisfacer las necesidades de los usuarios y del negocio en diferentes contextos de uso. Estos atributos son esenciales para evaluar y garantizar la calidad tanto del producto como de su uso, proporcionando un marco para medir y mejorar aspectos como la funcionalidad, la eficiencia, la seguridad y la mantenibilidad.

El estándar ISO/IEC 25010:2023 clasifica los atributos de calidad en dos modelos: el modelo de calidad del producto, que abarca características como adecuación funcional, rendimiento, compatibilidad, fiabilidad, usabilidad, seguridad, mantenibilidad y portabilidad, y el modelo de calidad en uso, que incluye efectividad, eficiencia, satisfacción, reducción de riesgos y cobertura de contexto. Estos atributos permiten diseñar y evaluar sistemas que sean confiables, seguros y adaptables a diversos escenarios, asegurando que cumplan con las expectativas del negocio y los usuarios. (iso, 2023)

2.2.3.1. Rendimiento:

El rendimiento es la capacidad de un sistema para ejecutar sus funciones de manera eficiente, cumpliendo con los requisitos de tiempo de respuesta y capacidad de procesamiento. Es importante que el sistema pueda manejar grandes volúmenes de datos y transacciones sin comprometer su funcionalidad y calidad de servicio. (Martin, 2017)

Muchas de las decisiones de arquitectura son acerca del rendimiento del sistema y en ocasiones, muchas de esas decisiones afectan el rendimiento de tal manera que es difícil arreglarlo. Es difícil y subjetivo dar una definición de rendimiento ya que esta puede variar dependiendo de lo que el cliente espera del negocio, pero establecer métricas que permitan medirlo ayuda a determinar la viabilidad del sistema. (Fowler et al., 2002) Teniendo en cuenta esto, el rendimiento podría medirse de diferentes maneras, a continuación se mencionarán algunas:

- **Tiempo de respuesta:** Se refiere al tiempo que tarda el sistema en responder a una solicitud del usuario. Un tiempo de respuesta rápido es deseable para mejorar la experiencia del usuario y la eficiencia del sistema.
- **Latencia:** Es el tiempo que tarda un dato en transferirse de un punto a otro dentro del sistema. Una baja latencia es crucial para garantizar la eficiencia en la comunicación entre componentes y servicios del sistema.
- **Throughput:** Se refiere a la cantidad de operaciones que el sistema puede procesar en un período de tiempo determinado. Un alto throughput indica que el sistema puede manejar un gran volumen de operaciones de manera eficiente.
- **Utilización de recursos:** Fowler también hace referencia a la utilización de recursos como métrica para medir el rendimiento. Esto incluye el uso de CPU, memoria, almacenamiento y otros recursos del sistema. Mantener una utilización equilibrada de recursos es importante para evitar cuellos de botella y garantizar un rendimiento óptimo.
- **Escalabilidad:** Si bien no es una métrica específica, es un aspecto importante del rendimiento. La capacidad de escalar el sistema para manejar cargas de trabajo crecientes sin degradación significativa del rendimiento es fundamental en el diseño de arquitecturas eficientes. (Fowler et al., 2002)

2.2.3.2. Fiabilidad:

La fiabilidad es la capacidad del sistema para funcionar de manera correcta y predecible tanto en situaciones normales como adversas. La fiabilidad es esencial para la efectividad y satisfacción de un usuario en el sistema. Diseñar sistemas fiables significa que puedan resistir fallas y recuperarse de manera eficiente, garantizando que el sistema pueda seguir operando en todo momento. La fiabilidad incluye conceptos como la tolerancia a fallos, la capacidad de recuperación, la gestión adecuada de errores y la disponibilidad continua.

- **Tolerancia a fallos:** Es la capacidad que tiene el sistema para continuar funcionando en casos de fallos en sus componentes o en el entorno en el que opera. Esto implica diseñar el sistema de manera que pueda detectar, aislar y recuperarse de los fallos de forma automática o manual, sin que el usuario final experimente interrupciones significativas en el servicio. En términos prácticos, la tolerancia a fallos requiere estrategias y mecanismos como la redundancia de componentes críticos, la capacidad de conmutación por error (failover), el monitoreo constante

del estado del sistema, la recuperación automática de errores y la capacidad de continuar operando en un estado degradado o reducido sin afectar negativamente la experiencia del usuario.

- **Disponibilidad:** Es la capacidad de un sistema de estar operativo y accesible cuando los usuarios lo necesitan. En otras palabras, se trata de la medida en que un sistema está disponible para proporcionar sus servicios de manera continua y sin interrupciones no planificadas.

  La disponibilidad es un atributo de calidad fundamental en la arquitectura de software, ya que los usuarios esperan que los sistemas estén disponibles en todo momento para realizar sus tareas y procesos. Esto implica diseñar el sistema con redundancias, mecanismos de recuperación ante fallos y estrategias de tolerancia a fallos para minimizar el tiempo de inactividad y garantizar un alto nivel de disponibilidad.

  Para calcular la disponibilidad del sistema, se pueden utilizar métricas como el tiempo medio entre fallos (MTBF), el tiempo medio de reparación (MTTR) y la tasa de fallos (FR). Estas métricas se utilizan en combinación con estrategias de diseño como redundancia, tolerancia a fallos y sistemas de respaldo para mejorar la disponibilidad del sistema.

2.2.3.3. Interoperatividad:

Es la capacidad de diferentes sistemas o componentes para comunicarse, intercambiar datos con otros componentes internos o externos, y trabajar de manera conjunta de forma eficiente y sin inconvenientes. Esto implica que la interoperabilidad no solo abarca la capacidad de transferir datos, sino también de interpretarlos correctamente una vez intercambiados. (Bass et al., 2012)

La interoperabilidad es un principio clave en el diseño de arquitecturas de software. Es importante diseñar sistemas que puedan comunicarse con otros sistemas de manera efectiva y sin problemas, lo cual es fundamental para la integración y la colaboración entre componentes y servicios. (Martin, 2017)

Para lograr esto se pueden usar interfaces claras y contratos bien definidos que garanticen la integración entre diferentes módulos y que los componentes puedan intercambiar datos y funcionar de manera conjunta de manera eficiente, sin depender de detalles internos de implementación.

Al mantener una arquitectura flexible y modular, se promueve la interoperabilidad y se evitan acoplamientos excesivos, lo que favorece la evolución y el mantenimiento del sistema a lo largo del tiempo.

Para lograr una buena interoperabilidad, es importante el uso de estándares abiertos y protocolos de comunicación comunes que faciliten la integración y la interacción entre los diferentes componentes y sistemas. Esto implica diseñar interfaces claras y contratos bien definidos que permitan una comunicación sin problemas y una integración eficiente de los servicios ofrecidos por los sistemas.

Además, es importante considerar aspectos como la compatibilidad de versiones, la gestión de errores y la seguridad en el diseño de sistemas interoperables. Esto garantiza que los sistemas puedan

intercambiar datos y servicios de manera confiable y segura, promoviendo una mayor flexibilidad y adaptabilidad en el entorno empresarial.

2.2.4. Escenarios de calidad

Son descripciones narrativas que ilustran cómo se espera que el software se comporte en situaciones específicas que representan los diferentes aspectos de la calidad del software. Estos escenarios ayudan a identificar posibles problemas de calidad en las primeras etapas del desarrollo del software. (Bass et al., 2012)

- **Nombre del Escenario:** Una descripción clara y concisa del escenario que se está evaluando.
- **Fuente del Estímulo:** La entidad o sistema que genera el estímulo.
- **Estímulo:** El evento o acción que desencadena el escenario.
- **Entorno:** Las condiciones en las que ocurre el escenario, como la carga de trabajo, la disponibilidad de recursos, etc.
- **Artefactos de Soporte:** Los componentes o elementos del sistema que participan en el escenario.
- **Respuesta:** La reacción esperada del sistema al estímulo.
- **Medición de la Respuesta:** Cómo se cuantifica o evalúa la respuesta del sistema.

2.2.5. Patrones y tácticas de arquitectura

Las tácticas de arquitectura son decisiones de diseño específicas que se toman para satisfacer ciertos requisitos no funcionales de un sistema, permiten abordar aspectos como la fiabilidad, el rendimiento, la seguridad, la disponibilidad y otros atributos de calidad que son críticos para el éxito del sistema. Algunas de las tácticas que se pueden implementar son: la redundancia de componentes para mejorar la fiabilidad, el uso de cachés para mejorar el rendimiento y la implementación de cortafuegos para aumentar la seguridad.
Los patrones de arquitectura son enfoques o estrategias que permiten abordar problemas comunes y mejorar la calidad y la eficiencia de una arquitectura de software. Los patrones de arquitectura son plantillas reutilizables que describen cómo resolver problemas de diseño en un contexto específico y están más asociados a la estructura y el diseño general del sistema. Estos patrones encapsulan las decisiones de diseño exitosas y las mejores prácticas desarrolladas por expertos en el campo de la arquitectura de software (Bass et al., 2012). Según (Gamma et al., 1996), un patrón tiene 4 elementos esenciales:

- **Nombre del patrón:** Describe el problema de diseño, la solución y sus consecuencias un una o dos palabras
- **Problema:** Describe cuando se puede aplicar el patrón, el problema y el contexto.

- **Solución:** Describe los elementos que componen la solución. No es una implementación concreta ya que el patrón es una plantilla que se puede aplicar en diferentes situaciones
- **Consecuencias:** Resultados y las compensaciones de aplicar el patrón, es esencial para evaluar el patrón, costos y beneficios de la implementación.

Los patrones de diseño representan soluciones probadas y establecidas para problemas comunes en el diseño de software. Al aplicar estos patrones, se pueden aprovechar las mejores prácticas y evitar reinventar la rueda, lo que ahorra tiempo y esfuerzo en el desarrollo del proyecto. Es crucial garantizar que el sistema sea robusto, escalable, mantenible y cumpla con los requisitos funcionales y no funcionales establecidos. Esto es importante para la toma de decisiones técnicas, algunos de los patrones de diseño que se pueden tener en cuenta son:

Component-Based Architecture	
Problema:	Dificultad para mantener sistemas grandes y complejos.
Solución:	Descomponer el sistema en componentes reutilizables e independientes.
Consecuencias:	Facilita la modularidad, la reutilización y la evolución del sistema.

Tabla 2.1: Patrón Component-Based Architecture

Service-Based Architecture	
Problema:	Dificultad para distribuir la funcionalidad del sistema en diferentes máquinas.
Solución:	Descomponer el sistema en servicios independientes y autónomos.
Consecuencias:	Facilita la modularidad, la reutilización y la evolución del sistema.

Tabla 2.2: Patrón Service-Based Architecture

Model-View-Controller (MVC)	
Problema:	Dificultad para separar la lógica de la presentación del usuario.
Solución:	Dividir la interfaz de usuario en tres componentes: modelo, vista y controlador.
Consecuencias:	Mejora la modularidad, la reutilización del código y la facilidad de mantenimiento.

Tabla 2.3: Patrón Model-View-Controller (MVC)

Layers	
Problema:	Dificultad para organizar un sistema complejo en componentes modulares.
Solución:	Descomponer el sistema en capas con diferentes responsabilidades.
Consecuencias:	Mejora la modularidad, la reutilización del código y la facilidad de mantenimiento.

Tabla 2.4: Patrón Layers

Microservices	
Problema:	Dificultad para desarrollar y mantener sistemas monolíticos grandes.
Solución:	Descomponer el sistema en pequeños servicios independientes.
Consecuencias:	Mejora la escalabilidad, la flexibilidad y la velocidad de desarrollo.

Tabla 2.5: Patrón Microservices

Middleware	
Problema:	Dificultad para proporcionar servicios comunes a diferentes aplicaciones.
Solución:	Introducir una capa de software que se sitúa entre las aplicaciones y los recursos de red.
Consecuencias:	Mejora la modularidad, la reusabilidad y la separación de preocupaciones del sistema.

Tabla 2.6: Patrón Middleware

Event-Driven Architecture (EDA)	
Problema:	Dificultad para coordinar la comunicación entre componentes loosely coupled.
Solución:	Basar la comunicación del sistema en la publicación y suscripción a eventos.
Consecuencias:	Mejora la escalabilidad, la flexibilidad y la desacoplación de los componentes del sistema.

Tabla 2.7: Event-Driven Architecture (EDA)

Hexagonal Architecture	
Problema:	Dificultad para mantener la lógica de negocio independiente de las tecnologías específicas.
Solución:	Definir una capa central de lógica de negocio independiente de la capa de presentación y la capa de persistencia.
Consecuencias:	Mejora la testabilidad, la reusabilidad y la mantenibilidad de la lógica de negocio.

Tabla 2.8: Patrón Hexagonal Architecture

Publish/Subscribe Architecture	
Problema:	Dificultad para coordinar la comunicación entre muchos componentes que necesitan recibir la misma información.
Solución:	Utilizar un mecanismo de publicación y suscripción para que los componentes interesados puedan recibir mensajes.
Consecuencias:	Mejora la escalabilidad y la desacoplación de los componentes del sistema.

Tabla 2.9: Patrón Publish/Subscribe Architecture

Las tácticas están diseñadas para abordar un aspecto particular relacionado con un atributo de calidad, como el rendimiento, la fiabilidad o la seguridad. Estas tácticas pueden ser utilizadas para optimizar o mejorar aspectos específicos de la arquitectura, permitiendo así lograr los objetivos de calidad establecidos para el sistema. (Bass et al., 2012)

Repository Pattern	
Problema:	Dificultad para abstraer el acceso a los datos del resto de la aplicación.
Solución:	Introducir una capa de abstracción que encapsule el acceso a los datos.
Consecuencias:	Mejora la modularidad, la reusabilidad, la testabilidad y la ocultación de la complejidad del acceso a datos.

Tabla 2.10: Patrón repository

2.2.5.1. Tácticas de disponibilidad

- **Component Redundancy (Redundancia de componentes):** Táctica que implica tener múltiples copias de componentes críticos del sistema para garantizar la continuidad de la operación en caso de fallo de uno de ellos.

- **Monitor:**consiste en la supervisión continua de los componentes de un sistema para detectar posibles fallos o degradaciones en su funcionamiento. Este monitoreo puede abarcar aspectos como el rendimiento, la disponibilidad, la utilización de recursos, entre otros. Al detectar alguna anomalía, el sistema puede tomar medidas automáticas para mitigar el problema o notificar a los operadores para que intervengan manualmente. Esta táctica es fundamental para mantener la disponibilidad y el rendimiento de un sistema en niveles óptimos y para prevenir interrupciones inesperadas.

- **Exception detection (Detección de excepciones):** Se centra en la detección temprana de excepciones o errores en el sistema. Esta táctica implica la implementación de mecanismos y técnicas que permiten identificar y capturar excepciones cuando ocurren durante la ejecución del sistema.

  La detección de excepciones es esencial para garantizar la fiabilidad y la robustez del software, ya que permite manejar situaciones inesperadas de manera adecuada, evitando que los errores se propaguen y afecten negativamente la experiencia del usuario o la funcionalidad del sistema.

  Al aplicar la táctica de "exception detection", se pueden implementar estrategias como la validación de entradas, el uso de bloques try-catch para capturar excepciones específicas, el registro de errores para su posterior análisis y corrección, entre otras técnicas que contribuyen a mejorar la calidad y la confiabilidad del software.

- **Retry:** La táctica "retry"se refiere a la estrategia de reintentar una operación o acción después de que ha fallado debido a una condición temporal o intermitente. Esta táctica se utiliza para mejorar la confiabilidad y la tolerancia a fallos de un sistema al proporcionar una forma de recuperarse automáticamente de errores transitorios.

  Cuando se implementa la táctica retry", el sistema realiza automáticamente un nuevo intento de ejecutar la operación que falló, generalmente después de un breve período de tiempo. Esta

táctica se aplica especialmente en situaciones donde se espera que el problema que causó el fallo sea temporal y pueda resolverse por sí solo en un corto período.

- **Resilience (Capacidad de recuperación):** Diseñar el sistema para que pueda recuperarse de manera automática y rápida ante fallas o interrupciones.
- **Load Balancing (Balanceo de carga):** Distribuir la carga de trabajo entre múltiples instancias o servidores para evitar la saturación y mantener la disponibilidad del sistema.
- **Data Replication (Replicación de datos):** Mantener copias de datos en diferentes ubicaciones para garantizar el acceso a la información en caso de fallos locales.
- **Scalability (Escalabilidad):** Diseñar el sistema para escalar horizontal o verticalmente según sea necesario para mantener el rendimiento y la disponibilidad.
- **Caching:** Utilizar cachés para almacenar datos temporalmente y reducir el tiempo de acceso, aliviando la carga en los recursos principales del sistema.

2.2.5.2. Tácticas de interoperabilidad

- **Orchestrate (Orquestar):** Esta táctica implica la coordinación y gestión centralizada de múltiples servicios y sistemas para garantizar una interacción coherente y eficiente. Se utilizan mecanismos como la composición de servicios, los flujos de trabajo y las reglas de negocio para orquestar la ejecución de procesos distribuidos y complejos.
- **Discover Service (Descubrir Servicio):** Esta táctica se centra en la capacidad de encontrar y acceder dinámicamente a servicios disponibles en un entorno distribuido. Se utilizan registros de servicios, directorios de servicios o mecanismos de descubrimiento automático para identificar y utilizar servicios de manera flexible y sin dependencias estáticas.
- **Tailor Interface (Adaptar Interfaz):** Esta táctica se refiere a la capacidad de adaptar las interfaces de los servicios o sistemas para que se ajusten a los requisitos específicos de cada contexto de uso. Se utilizan mecanismos de adaptación como la personalización de interfaces, la configuración dinámica de opciones y la entrega de contenido adaptado para proporcionar experiencias de usuario personalizadas y eficientes.

2.2.5.3. Tácticas de rendimiento

- **Introduce Concurrencia (Introduce Concurrency):** Esta táctica implica la introducción de procesos o hilos concurrentes para manejar tareas de manera paralela. Al permitir que varias operaciones se ejecuten simultáneamente, se puede aumentar la eficiencia y reducir los tiempos de respuesta en el sistema. Esta táctica es útil para aplicaciones que tienen tareas independientes que pueden ejecutarse en paralelo sin afectar la coherencia de los datos.

- **Scheduling Policies (Políticas de Planificación):** Esta táctica se refiere a la elección de políticas de planificación adecuadas para gestionar la asignación de recursos y la ejecución de tareas en un sistema. Las políticas de planificación pueden incluir algoritmos como el planificador de lotería, planificador FIFO (First In, First Out), planificador de prioridades, entre otros. La selección de la política de planificación adecuada puede mejorar el rendimiento al optimizar la asignación de recursos y la ejecución de tareas en el sistema.

2.3. Evaluación de arquitecturas de software

La evaluación de arquitectura de software es un proceso crítico que permite analizar y validar la estructura y el diseño de un sistema software (Garlan and Shaw, 1993). Consiste en examinar diferentes aspectos de la arquitectura, como su capacidad para cumplir con los requisitos funcionales y no funcionales, su coherencia interna, su adaptabilidad a cambios futuros, entre otros.
Evaluar la arquitectura de software es fundamental por varias razones. En primer lugar, permite garantizar la calidad del sistema al verificar que cumpla con los estándares y requisitos establecidos (Bass et al., 2012). Además, ayuda a identificar posibles riesgos y problemas en la arquitectura, lo que permite abordarlos a tiempo y minimizar su impacto en el desarrollo y la implementación del sistema.

2.3.0.1. SAAM (Software Architecture Analysis Method)

Desarrollada por AT&T Bell Laboratories. SAAM se centra en identificar y analizar los atributos de calidad de la arquitectura, como la modularidad, la reutilización, la escalabilidad y la flexibilidad (Garlan and Shaw, 1993). Utiliza técnicas de revisión y análisis de escenarios para evaluar la capacidad de la arquitectura para satisfacer los requisitos del sistema y las necesidades del usuario.

2.3.0.2. ATAM (Architecture Tradeoff Analysis Method)

Desarrollada por el Software Engineering Institute (SEI) (Kazman et al., 2000). ATAM se enfoca en identificar y evaluar los trade-offs o compromisos en la arquitectura, ayudando a tomar decisiones sobre posibles mejoras y ajustes para optimizar el rendimiento y la eficiencia del sistema.

2.3.0.3. ARID (Architecture Review and Improvement Method)

Se enfoca en la revisión continua de la arquitectura para identificar oportunidades de mejora y optimización (Garlan and Shaw, 1993). ARID proporciona un marco de trabajo estructurado para evaluar la arquitectura en diferentes etapas del ciclo de vida del software y garantizar su evolución y adaptación a los cambios del entorno.

2.4. Metodología

2.4.1. Scrum

Scrum es un marco de trabajo ágil utilizado en el desarrollo de software y proyectos complejos utilizado para organizar el trabajo en ciclos cortos llamados sprints, se enfoca en la entrega de productos que agreguen valor al cliente de manera iterativa e incremental, y en la colaboración efectiva entre equipos multidisciplinarios (Schwaber and Sutherland, 2017). Se basa en principios como la transparencia, la inspección y la mejora continua, y utiliza ceremonias como reuniones diarias (Daily Scrum), revisiones de sprint (Sprint Review) y retrospectivas (Sprint Retrospective) para gestionar el trabajo de manera eficiente y mejorar continuamente el proceso (Judicial, 2023). En el contexto de Redjudicial, el uso de scrum como metodología permite una gestión flexible y adaptativa del proyecto, ajustar prioridades y enfocarse en las necesidades del negocio. Como metodología iterativa e incremental, permite liberar funcionalidades para ser validadas por el negocio y tener retroalimentación continua.

- **Roles:** Siguiendo el marco de scrum se deben definir responsabilidades agrupadas en roles como el Scrum Master, que facilita el proceso y elimina obstáculos, el Product Owner, responsable de definir y priorizar las funcionalidades del producto, y el Equipo de Desarrollo, encargado de implementar las funcionalidades (Schwaber and Sutherland, 2017).
- **Seguimiento y planificación:** Scrum utiliza artefactos como el Product Backlog, que contiene las funcionalidades pendientes, y el Sprint Backlog, que define las tareas para un período (sprint) corto. Esto permite un seguimiento transparente del progreso y una planificación adaptativa (Cohn, 2010).
- **Ceremonias:** Se realizan reuniones diarias (Daily Scrum) para compartir avances y coordinar el trabajo, revisiones de sprint (Sprint Review) para demostrar el trabajo completado y obtener retroalimentación, y retrospectivas (Sprint Retrospective) para identificar mejoras en el proceso (Sutherland and Altman, 2014)

2.4.2. As-Is To-Be

La metodología As-Is To-Be es una técnica utilizada en la gestión de procesos para analizar y mejorar la situación de un proceso considerando su estado actual (As-Is) y diseñar la situación futura deseada (To-Be) (Jones, 2012). Este enfoque permite identificar las brechas entre el estado actual y el estado deseado de un proceso y define acciones para cerrar esas brechas y lograr la transformación.

Al ser aplicada en proyectos de arquitectura y desarrollo de software como el proyecto Redjudicial se enfoca en comprender y documentar el estado actual (As-Is) de los sistemas, aplicaciones y procesos existentes dentro de Redjudicial. Esto incluye la identificación de componentes, infraestructura tecnológica, flujos de datos, interfaces y cualquier otra característica del sistema (Garlan et al., 2004).

Algunos aspectos clave de la aplicación de la metodología As-Is incluyen:

- **Análisis del estado actual:** Análisis detallado de la arquitectura del sistema, bases de datos, integraciones, APIs, y otros componentes técnicos para comprender su funcionamiento y relaciones (Bass et al., 2012).
- **Documentación exhaustiva:** Es fundamental generar documentación detallada y precisa con toda la información recopilada durante el análisis, utilizando herramientas como diagramas de arquitectura, mapas de procesos, modelos de datos, entre otros (Rozanski and Woods, 2011).
- **Identificación de puntos críticos y áreas de mejora:** A través del análisis As-Is, se pueden identificar problemas de rendimiento, redundancias, inconsistencias, y otros aspectos que necesitan ser mejorados o corregidos.
- **Evaluación de tecnologías y herramientas:** También se puede evaluar la idoneidad de las tecnologías y herramientas utilizadas en el contexto actual, identificando oportunidades para modernizar o adoptar nuevas tecnologías que mejoren la eficiencia y la calidad del software.

2.4.3. Software Architecture Analysis Method (SAAM)

El Software Architecture Analysis Method (SAAM) es una técnica utilizada para evaluar la calidad y efectividad de la arquitectura de software, centrándose en atributos de calidad a través de escenarios que son evaluados y comparados contra los reales (Ionita et al., 2002). SAAM es útil para evaluar rápidamente muchos atributos de calidad como la modificabilidad , la portabilidad, la extensibilidad, entre otros. El método también se puede utilizar para evaluar los aspectos de calidad de las arquitecturas de software como el rendimiento o la fiabilidad en los cuales se centra este proyecto.

Este método se puede emplear para desde diferentes enfoques dependiendo del número de arquitecturas a evaluar, por ejemplo, comparando entre dos arquitecturas. En el caso de analizar una arquitectura, SAAM indica los puntos débiles o los puntos fuertes, junto con los puntos donde la arquitectura falla en satisfacer los requerimientos de modificabilidad.

SAAM es una metodología iterativa que cuenta con siete pasos:

1. **Identificación de escenarios de uso:** En este paso, se identifican los escenarios de uso o situaciones en las que la arquitectura del software será probada. Estos escenarios representan las diferentes formas en que los usuarios interactuarán con el sistema.
2. **Descomposición de escenarios:** Los escenarios de uso se descomponen en pasos más detallados que describen las acciones específicas que los usuarios realizarán en cada escenario.
3. **Identificación de elementos arquitectónicos:** Se identifican los elementos clave de la arquitectura del software, como componentes, módulos, capas, etc., que están involucrados en cada paso de los escenarios de uso.
4. **Asignación de atributos de calidad:** Se asignan atributos de calidad o características de rendimiento, seguridad, usabilidad, etc., a los elementos arquitectónicos identificados. Estos

atributos ayudan a evaluar la capacidad de la arquitectura para cumplir con los requisitos y expectativas del sistema.

5. **Evaluación de impacto:** Se evalúa el impacto de cada paso de los escenarios de uso en los atributos de calidad asignados a los elementos arquitectónicos. Esto ayuda a identificar áreas de fortaleza y debilidad en la arquitectura.

6. **Análisis de alternativas:** Se exploran y evalúan diferentes alternativas arquitectónicas para mejorar los atributos de calidad identificados como críticos o problemáticos durante la evaluación de impacto.

7. **Toma de decisiones:** Con base en los resultados del análisis, se toman decisiones informadas para realizar mejoras en la arquitectura del software y abordar los problemas identificados.

Para este proyecto, SAAM es una metodología conveniente porque proporciona un enfoque estructurado y sistemático para evaluar la arquitectura del sistema, identificar áreas de mejora y tomar decisiones informadas. Al combinar SAAM con el enfoque As-Is To-Be, permite una visión completa de la arquitectura actual y futura, lo que facilita la alineación con los objetivos del negocio, además de que permite considerar las alternativas arquitectónicas para asegurar que la arquitectura evolucione para satisfacer los objetivos del negocio.

2.4.4. DevOps

DevOps es un conjunto de prácticas que combinan el desarrollo de software (Dev) y las operaciones de TI (Ops) con el objetivo de acortar el ciclo de vida del desarrollo del sistema y proporcionar una entrega continua con alta calidad de software. DevOps promueve una cultura de colaboración entre equipos que tradicionalmente funcionaban en silos, como desarrollo, operaciones y calidad. Estas prácticas, respaldadas por la automatización y las herramientas de integración continua e implementación continua (CI/CD), mejoran la eficiencia, velocidad y confiabilidad del desarrollo y la entrega de software. (Kim et al., 2016).
DevOps se puede considerar como una cultura que promueve la colaboración entre los equipos de desarrollo y operaciones, donde ambos son responsables de la entrega del software al cliente. La automatización de la construcción, despliegue y pruebas del software es importante para DevOps, ya que permite cumplir con la entrega del software.

2.4.4.1. Prácticas de DevOps

- **Integración Continua (CI):** Automatiza la integración del código de todos los desarrolladores en un solo proyecto de software.
- **Entrega Continua (CD):** Extiende la integración continua al automatizar la entrega de las aplicaciones a los entornos de pruebas y producción.
- **Monitoreo y Logging:** Monitoriza el rendimiento del software y captura datos de logs para la identificación y resolución rápida de problemas.

- **Infraestructura como Código (IaC):** Gestiona y aprovisiona la infraestructura a través de archivos de definición en lugar de procesos manuales.

2.4.4.2. Ventajas de DevOps

- **Mayor Frecuencia de Implementación:** Las organizaciones pueden liberar nuevas funcionalidades y actualizaciones más rápidamente.
- **Menos Fallos de Implementación:** La automatización y las pruebas continuas reducen los errores y problemas en las nuevas implementaciones.
- **Recuperación Rápida:** En caso de problemas, los equipos pueden identificar y solucionar los problemas rápidamente.
- **Mejor Colaboración:** Fomenta una cultura de colaboración y responsabilidad compartida entre los equipos de desarrollo y operaciones.

2.5. Estado del Arte

Desde el inicio de su operación, Redjudicial ha hecho implementaciones de software con el fin de automatizar el proceso de extracción de datos de Redjudicial.
Es difícil abordar métodos o técnicas utilizadas para el caso de uso de Redjudicial ya que, aunque la extracción de datos es un proceso común, el uso de estas técnicas aplicadas a Rama judicial no se ha documentado y no ha sido posible encontrar bibliografía publicada. Aunque partiendo de la premisa de que existen otras empresas que ofrecen el mismo servicio, se puede asumir que, de alguna manera, se está haciendo (posiblemente de forma manual).
Es por eso que se han recopilado datos de proyectos similares, teniendo en cuenta la fuente de datos y las necesidades planteadas para el caso de Redjudicial, como lo es la tolerancia a fallos y el procesamiento de grandes cantidades de datos, además de frameworks que actualmente son usados para la construcción de sistemas distribuidos, escalables y tolerantes a fallos.

2.5.0.1. Antecedentes

Redjudicial ha realizado algunos desarrollos de software para automatizar la sincronización de procesos judiciales y así reducir los errores y la lentitud presentada en el proceso. El principal de estos desarrollos consistió en un script en python que, usando la librería selenium, extraía los datos a través de Webscraping. Gracias a esta aplicación se podían extraer los datos de Ramajudicial; no obstante, el proceso era lento, el consumo de recursos era alto y la gestión de errores era difícil, y aunque el proceso podía extraer la información, apenas podía procesar 500 procesos al día, lo que no era eficiente ni satisfacía las necesidades de Redjudicial.
Posteriormente se implementó una aplicación en .NET Framework que utilizaba Selenium para extraer la información, este aplicativo utilizaba paralelismo, pero el manejo de recursos al usar Chrome no era eficiente, la disponibilidad de Rama Judicial se tornaba inestable y esto inducía al

sistema a errores, por lo cual el sistema constantemente dejaba de estar disponible y, aunque su velocidad mejoró con respecto a la versión en Python, este solo era capaz de procesar hasta 3.000 procesos por día, lo cual, aunque es un incremento considerable, no es suficiente para satisfacer las necesidades de la operación de Redjudicial.

2.5.0.2. Extracción de datos y análisis sentimental de "Twitter" usando scrapping

Durante las elecciones presidenciales del 2016 en Estados Unidos, se hizo una investigación sobre las emociones de las personas en la red social Twitter. Para este caso, se extrajeron comentarios de las redes a través de web scraping y se etiquetaron para luego ser procesados. (Jain et al., 2019)

2.5.0.3. Un método completo y rápido de scrapping para recolectar Tweets

Aunque se puede considerar que el scrapping es un método de extracción de datos bastante lento comparado a las APIs, en este caso de estudio se consideró como una mejor alternativa ya que el API de Twitter incluye restricciones del número de tweets que se pueden obtener por petición y del número de peticiones que se pueden hacer por día incluso en versiones premium del API. Al implementar paralelismo en el proceso de extracción, los resultados del sistema mejoran considerablemente. En los resultados de esta implementación se obtuvo que, mientras una implementación monohilo requería 30.576 segundos para rastrear 222.194 tweets, lo cual representa una velocidad de 7,26 tweets/seg. Por el contrario, el multiproceso de DeepScrap muestra 21,61 tweets/seg, es decir, sólo necesita 10.282 segundos para la misma cantidad de tweets, lo cual representa 33 % del tiempo requerido en un solo hilo y con recursos hardware promedio.

Aunque esta solución podría presentar una gran mejora para Redjudicial, es necesario tener en cuenta que el aplicativo Ramajudicial no cuenta con una gran capacidad para procesar múltiples solicitudes en comparación con Twitter. (You et al., 2021)

2.5.0.4. The Billion Prices Project: Using Online Prices for Measurement and Research

El "Billion Prices Project" es un proyecto de investigación desarrollado por el MIT Sloan School of Management en colaboración con la Universidad de Harvard. Su objetivo principal es rastrear y analizar los cambios en los precios de bienes y servicios en tiempo real a nivel mundial. El proyecto utiliza técnicas de web scraping (extracción de datos de sitios web) para recopilar una gran cantidad de información sobre precios de productos y servicios en línea. Este proyecto es especialmente valioso porque proporciona una fuente de datos en tiempo real que puede utilizarse para analizar la inflación, evaluar tendencias económicas y realizar un seguimiento de los cambios en los precios de bienes y servicios en todo el mundo. La información recopilada se utiliza tanto con fines académicos como para comprender mejor la economía global y tomar decisiones basadas en datos. Para la construcción de los robots scraping se utilizó una herramienta de web scraping que permitía usar el mouse para iniciar el proceso a seguir y, finalmente, generar el robot. El uso de scraping permitió obtener información en tiempo real y en gran volumen desde diferentes regiones

geográficas. En este caso de estudio no se abordan problemas como los cambios en los sitios web y el mantenimiento de los scripts de extracción, ya que se cubrieron pocos proveedores y pocas categorías. (Cavallo and Rigobon, 2016)

2.5.0.5. An Implementation and Performance Evaluation of Fast Web Crawler with Python

En este proyecto, se lleva a cabo la implementación del web scraping para la extracción de datos de diversas páginas web. El objetivo principal es analizar los efectos de la implementación de una aplicación multi-hilo en el rendimiento del web scraping. Como resultado, se observó que la implementación de múltiples hilos mejora significativamente la velocidad de extracción de datos en comparación con las implementaciones de un solo hilo. (Kim, 2019)

2.5.1. Comparación

Para este proyecto se tomaron como referencia proyectos en los cuales se aborda la extracción de datos desde sitios web, siendo este el principal objetivo del sistema de Redjudicial; no obstante, también se ha identificado la necesidad de que el proceso sea rápido, ya que Redjudicial debe obtener información de todos los procesos judiciales que tiene a su cargo (aproximadamente 16.000 en la actualidad) y además garantizar tolerancia a los fallos que se puedan presentar en el aplicativo ramajudicial.

El uso de scraping y paralelismo se plantean como opciones viables para aumentar la capacidad de sincronización de procesos judiciales del negocio, además de que garantizan que la información que se obtiene es fiable (la fuente de datos es fiable), no obstante, actualmente el aumento en la capacidad sigue estando limitado por la cantidad de recursos del sistema (hardware) y no asegura la gestión de los errores que pueden llegar a detener la sincronización. Es por eso que la solución propuesta debe enfocarse en escalabilidad y tolerancia a fallos.

Tabla 2.11: Trabajos previos VS propuesta actual

Proyecto anterior	Propuesta actual
Extracción de datos y análisis sentimental de "Twitter" usando scrapping: En este proyecto se abordó la extracción de datos usando scrapping para descargar información de Twitter	Se aborda la extracción de datos y además tolerancia a fallos ya que Twitter tiene estabilidad y alta disponibilidad a diferencia de rama judicial
Deepcrapping, Un método completo y rápido de scrapping para recolectar Tweets En este proyecto se abordó la extracción de datos usando scrapping y además se usa paralelismo para que el proceso sea más rápido aprovechando mejor los recursos hardware	Se aborda la extracción de datos pero además se aprovecharán los recursos computacionales de múltiples nodos
Temporal.io Es un proyecto que se enfoca en el flujo del proceso, el encolamiento y la tolerancia a fallos, al ser una herramienta no aborda un objetivo específico	Podría considerar una aplicación de las funcionalidades y características ofrecidas por temporal, agregando la función de extracción de datos
The Billion Prices Project: Using Online Prices for Measurement and Research Aborda la extración de datos usando scraping desde diferentes sitios web pero no resuelve los problemas de tolerancia a fallos y el procesamiento de grandes cantidades de datos	Tiene como objetivo la extracción de datos pero requiere garantizar tolerancia a fallos y el provesamiento de datos.
An Implementation and Performance Evaluation of Fast Web Crawler with Python En este proyecto no solo se aborda la extracción de datos sino que también se tiene en cuenta la necesidad de mejorar el rendimiento de este a través de una arquitectura multihilo	Para la extracción de datos desde ramajudicial, Redjudicial ha planteado la necesidad de garantizar que los datos se extraigan con rapidez y además que pueda tolerar los fallos que se presentan en rama judicial.

Capítulo 3

Desarrollo del Proyecto

3.1. Contexto del negocio

REDJUDICIAL SAS y SISTEMA JUDICIAL WEB SAS prestan un servicio integral para Despachos de Abogados y Departamentos Jurídicos de Empresas y Entidades Públicas, siendo Redjudicial pionera a nivel nacional en el uso de la tecnología de Internet para la Vigilancia y Control de Procesos Judiciales, minimizando tiempo y costos a sus más de 1.200 usuarios.

Sistema judicial nace como una herramienta virtual para complementar los servicios de seguimiento de los procesos con fotos de las providencias, estudios y depuración de listas de expedientes, administración y actualización de la información y el manejo de la misma en la nube, en nuestro Software de gestión de procesos, con manejo de clientes, personal, gestión documental, ruta procesal, agenda y auditoría, historial de movimientos y mucho más.

3.1.1. Productos y servicios

Redjudicial ofrece productos y servicios tecnológicos a sus clientes, plataformas a través de las cuales pueden gestionar los procesos judiciales o recibir información de los mismos.

- **Sistema Judicial:** El software de Sistema Judicial está diseñado para que los abogados puedan tener una oficina virtual, permitiéndoles gestionar sus expedientes, clientes, personal de trabajo y visualizar los movimientos de sus procesos a través de Redjudicial y/o Redjudicial Premium. Sistema Judicial ofrece diversas funcionalidades, tales como la creación de informes de procesos, manejo de agendas y rutas procesales, entre otras.
  Sistema Judicial se creó con el propósito de que los abogados tengan acceso a cada documento relacionado con sus procesos desde cualquier lugar, evitando inconvenientes como la pérdida de documentos físicos. La digitalización de documentos asegura su integridad y evita daños por deterioro, etc.
  Además, Sistema Judicial permite auditar y supervisar no solo los procesos, sino también el trabajo del personal mediante un Log de Acciones, en el cual se registra cada cambio realizado. Este software, en conjunto con Redjudicial, busca hacer la labor del abogado más cómoda y eficiente.

- **RedJudicial:** Es un servicio de vigilancia diseñado para monitorear los movimientos diarios en los distintos despachos judiciales disponibles en los micrositios de Rama Judicial, tales como TYBA, SAMAI, RAMA, Consejo de Estado, publicaciones procesales, consulta de procesos nacional unificada. Este servicio permite detectar diversos movimientos como estados,

traslados, edictos, sentencias y avisos.
Una vez identificados los movimientos, se registran en el sistema utilizando el número de radicado de 9 dígitos. Además, se especifica el origen del proceso (tribunal, juzgado de ejecución, etc.), la actuación correspondiente, y los nombres del demandante y del demandado según lo publicado por el juzgado. Cabe destacar que la cantidad de demandados puede variar dependiendo de cómo los haya registrado el despacho.
El sistema de Redjudicial compara los datos registrados con la información que los clientes han añadido en su perfil de Redjudicial y remite los movimientos pertinentes a cada perfil.

- **Redjudicial premium:** Redjudicial Premium ofrece un servicio de vigilancia personalizada, enfocándose específicamente en los movimientos del proceso del cliente, en lugar de los movimientos generales del despacho. Este servicio se realiza ingresando en la sección de Consulta de Procesos y revisando el proceso y sus notificaciones mediante el número de radicado de 23 dígitos.

  Una ventaja significativa de Redjudicial Premium es la capacidad de acceder a las constancias secretariales, las cuales no se muestran en los estados generales y solo se pueden visualizar desde la Consulta de Procesos. Además, si los responsables de la Consulta de Procesos suben la información a tiempo, los usuarios pueden anticiparse y ganar un día más de término, obteniendo la notificación antes de que se fije oficialmente.

3.1.2. Estructura tecnológica

La estructura tecnológica de Redjudicial se compone de diversos elementos que abarcan desde la infraestructura hasta las prácticas y el personal involucrado.
En cuanto a la infraestructura, la empresa cuenta con una combinación de servidores on premise ubicados en sus instalaciones y servidores en la nube. Esta combinación le permite tener disponibles y accesibles los servicios que son para uso del público y aislados aquellos que no por seguridad.
En el ámbito de las aplicaciones, Redjudicial utiliza una variedad de software especializado para gestionar sus procesos y ofrecer sus servicios. Esto incluye aplicaciones desarrolladas internamente para las funciones específicas del negocio, así como soluciones de terceros para áreas como la gestión de proyectos, la colaboración en equipo y el análisis de datos y otras de terceros como Ramajudicial. Estas aplicaciones se integran de manera eficiente para garantizar la operación sin problemas de los diferentes procesos y actividades de la empresa.
La estructura de personal de tecnología incluye un director que supervisa las operaciones y la estrategia, personal de soporte técnico para resolver problemas y garantizar el funcionamiento de los sistemas, y un equipo de desarrollo que terceriza el desarrollo del extractor de datos pero mantiene un desarrollador interno para otros productos. Esta distribución permite enfocar recursos y talento de manera eficiente, asegurando calidad y eficiencia en el desarrollo de las soluciones.

Las prácticas de Redjudicial están enfocadas en la seguridad de la información y la protección de datos mediante procesos de cumplimiento y medidas de seguridad.

3.1.3. Objetivos del negocio

Es esencial definir claramente los objetivos del negocio para guiar el desarrollo y la implementación del sistema. Los objetivos del negocio establecen las metas estratégicas que la organización busca alcanzar a través de este proyecto, proporcionando una dirección clara y criterios de éxito medibles. Estos objetivos no solo guían el desarrollo técnico, sino que también aseguran que todas las actividades y decisiones del proyecto estén alineadas con las prioridades y necesidades de la organización.

Los objetivos del negocio de Redjudicial fueron definidos por las directivas de la empresa considerando tanto la eficiencia operativa como la satisfacción del cliente. Estos objetivos incluyen la mejora de la precisión y la rapidez en la actualización de procesos judiciales, la optimización de recursos tecnológicos y humanos, y la capacidad de escalar las operaciones en respuesta al crecimiento del negocio. Establecer estos objetivos permitió enfocarse en los resultados deseados y evaluar el éxito del proyecto de manera objetiva.

En la tabla 3.1 se describen los objetivos del negocio, junto con una descripción y los indicadores de éxito correspondientes para cada uno.

Objetivo	**Descripción**	**Indicador de Éxito**
Mejorar la Eficiencia Operativa	Reducir el tiempo y esfuerzo necesario para consultar y actualizar los procesos judiciales.	El Proceso debe automatizarse al 100 % de tal manera que la actualización de procesos judiciales no requiera intervención del personal.
Aumentar la Confiabilidad del Sistema	Incrementar la fiabilidad del sistema, minimizando los errores y fallos durante las consultas y actualizaciones.	Reducir la tasa de errores en las consultas y actualizaciones a menos del 5 %.
Asegurar el crecimiento de la operación	Asegurar que el sistema pueda manejar el aumento en la carga de trabajo sin comprometer la integridad de los datos o la eficiencia del sistema.	Capacidad de sincronizar hasta 20.000 procesos judiciales diarios (inicialmente) sin deterioro significativo en el rendimiento.
Aumento en la Precisión de las Notificaciones	La precisión de la información y su notificación al usuario es parte de la propuesta de valor, por eso el sistema debe garantizar que se haga a tiempo.	Puntualidad de las notificaciones. el retraso debe ser de máximo 1 día.

Tabla 3.1: Objetivos del negocio

3.1.4. Dominios del negocio

Para el contexto de Redjudicil se identifican 4 dominios principales, enfocados en los productos y servicios que ofrecen, pero este proyecto solo se centra en Premium.

- **Clientes:** Gestión de clientes desde el proceso de registro, la suscripción de servicios y la comunicación
- **Procesos judiciales:** Gestión de procesos judiciales incluye actividades de registro, sincronización, seguimiento, presentación.
- **Administración:** Gestión de la plataforma. Incluye actividades de acceso a usuarios y personal del negocio, parametrización
- **Suscripciones:** Gestión de la plataforma. Incluye actividades de acceso a usuarios y personal del negocio, parametrización

3.1.4.1. Subdominios

3.1.4.2. Lenguaje Ubicuo

El lenguaje ubicuo es un conjunto de términos y definiciones que se utilizan consistentemente en todo el proyecto para asegurar una comunicación asertiva y facilitar la colaboración.
El propósito del lenguaje ubicuo es crear una base sólida para la comunicación entre los desarrolladores, los interesados y todos los miembros del equipo del proyecto. Al utilizar los mismos términos con las mismas definiciones, se minimizan las ambigüedades y se mejora la precisión en la planificación, el desarrollo y la implementación del sistema.
Para desarrollar el lenguaje ubicuo de Redjudicial, se realizaron sesiones de trabajo colaborativo con los dueños del dominio y usuarios clave del sistema. Durante estas sesiones, se identificaron y definieron los términos esenciales relacionados con los procesos judiciales y los servicios ofrecidos por Redjudicial. El resultado es un conjunto de términos acordados que reflejan fielmente las necesidades y el contexto del proyecto.
En la tabla 3.2 se presentan los términos y sus respectivas definiciones, que constituyen el lenguaje ubicuo para el proyecto Redjudicial.

Término	Descripción
Proceso Judicial	Un conjunto de actos y actuaciones realizadas por los órganos jurisdiccionales (juzgados y tribunales) con el fin de resolver un conflicto o aplicar la ley en un caso específico.
Radicado	Número único de identificación asignado a cada proceso judicial en el sistema judicial colombiano.
Actuación	Acción o decisión registrada en un proceso judicial, como una sentencia, traslado, edicto, o aviso.
Rama Judicial	Entidad gubernamental responsable de la administración de justicia en Colombia, incluyendo los juzgados y tribunales.
Redjudicial	Plataforma que ofrece servicios de vigilancia y notificación de movimientos en procesos judiciales a los clientes.
Redjudicial Premium	Servicio avanzado de Redjudicial que ofrece vigilancia personalizada sobre movimientos específicos en procesos judiciales.
Sistema Judicial	Software diseñado para que los abogados gestionen sus expedientes, clientes, personal y visualicen los movimientos de sus procesos judiciales.
Consulta Procesos	Funcionalidad del Sistema Judicial que permite buscar y revisar el estado de un proceso judicial usando su número de radicado.
Log de Acciones	Registro de todas las acciones y cambios realizados dentro del Sistema Judicial.
Migración de Datos	Proceso de transferir datos de un sistema o entorno a otro, asegurando la integridad y disponibilidad de los datos.

Tabla 3.2: Lenguaje ubicuo

3.2. Gestión del proyecto

El seguimiento de las metodologías seleccionadas para el proyecto ha sido clave para su éxito. Desde el principio, se estableció una organización clara y eficiente que permitió avanzar de manera efectiva en cada etapa del desarrollo.

3.2.1. Planeación

Uno de los primeros pasos fue definir los roles y responsabilidades de cada miembro del equipo, asegurando así que todos tuvieran un propósito claro y contribuyeran de manera significativa al proyecto. Además, se implementaron acuerdos entre el negocio y el equipo de desarrollo acerca de la confidencialidad para proteger la información sensible del proyecto y garantizar la privacidad de los datos de los clientes y usuarios. Estos acuerdos fueron fundamentales para mantener la integridad

y la seguridad de la información en todo momento. En la tabla 3.3 se enumeran los roles que participaron durante la ejecución del proyecto.

Rol	Responsable	Funciones
Scrum Master	Royer David Estrada Esponda	El Scrum Master es el facilitador del equipo Scrum, asegurando que se sigan los principios y prácticas de Scrum. Su función principal es eliminar obstáculos, fomentar la comunicación efectiva y garantizar que el equipo alcance sus objetivos de entrega en cada sprint.
Arquitecto de software	Brian Mosquera	diseñar la arquitectura del sistema, asegurando que cumpla con los requisitos funcionales y no funcionales. Trabaja en estrecha colaboración con el equipo de desarrollo para garantizar la integridad técnica y la escalabilidad del sistema.
Product Owner (Negocio)	Rubén Pino	es el representante del cliente y el responsable de definir las funcionalidades y prioridades del producto. Trabaja en estrecha colaboración con el equipo de desarrollo para asegurar que se entreguen las características más valiosas para el cliente en cada iteración.
Product owner (Tech)	Alejandro Luna	Tiene un rol estratégico en la empresa, brindando orientación y dirección en la adopción de tecnologías y en la toma de decisiones técnicas importantes. Trabaja en conjunto con el arquitecto de software para validar las decisiones técnicas, proporcionar información técnica clave y asegurar la alineación de la estrategia tecnológica con los objetivos del negocio.
Desarrollador	Tercerizado.	El desarrollador de software es parte del equipo tercerizado encargado de implementar las funcionalidades y componentes del sistema según las especificaciones y diseños proporcionados por el arquitecto de software y el Product Owner. Su responsabilidad principal es escribir código limpio, eficiente y funcional, asegurando la calidad y el cumplimiento de los estándares de desarrollo establecidos.

Tabla 3.3: Roles del proyecto

En cuanto a la duración de los sprints, se estableció un calendario que permitió trabajar de manera iterativa y ágil, con sprints de dos semanas de duración. Esta metodología de trabajo en ciclos cortos permitió una mayor flexibilidad y adaptabilidad a los cambios y requerimientos del proyecto.
Las ceremonias del proyecto se llevaron a cabo de manera regular y sistemática, incluyendo reuniones de planificación de sprint, reuniones de revisión de sprint y retrospectivas que se hicieron cada

2 sprints; además, se omitieron las reuniones diarias. Estas ceremonias fueron fundamentales para mantener una comunicación fluida, identificar problemas y tomar decisiones informadas en cada etapa del proyecto.
Por último, el backlog del proyecto se gestionó de manera eficiente, priorizando las tareas y funcionalidades más importantes y asegurando un flujo de trabajo constante y productivo. Esto permitió cumplir con los objetivos establecidos en cada sprint y avanzar de manera progresiva hacia la meta final del proyecto.

En la tabla 3.4 se muestran las épicas que se definieron durante el proyecto y en la tabla 3.5 se enumeran las historias de usuario que se desarrollaron por cada épica.

ID	**Épica**	**Descripción**
RDJC-2	Diseño As-Is	Esta épica se centra en el análisis exhaustivo del diseño actual del sistema Redjudicial. Incluye la identificación de componentes, flujos de datos, interacciones entre módulos y su arquitectura general.
RDJC-19	Diseño To-Be	En esta épica, se define el diseño futuro deseado para el sistema Redjudicial. Se incluyen los cambios, mejoras y nuevas funcionalidades que se planean implementar para alcanzar los objetivos del proyecto.
RDJC-20	Plan de transición	El Plan de Transición abarca las actividades y acciones necesarias para llevar a cabo la migración del diseño As-Is al diseño To-Be de manera efectiva y sin interrupciones significativas en las operaciones.
RDJC-4	Desarrollo e implementación	Esta épica abarca todas las actividades relacionadas con la construcción y puesta en marcha del sistema Redjudicial basado en el diseño To-Be definido previamente. Además incluirá los ajustes necesarios para pruebas de concepto
RDJC-21	Evaluación	Esta épica se enfoca en la evaluación continua del proyecto Redjudicial en términos de cumplimiento de requisitos, calidad del sistema, rendimiento y satisfacción del cliente. Incluye la definición de métricas de evaluación, la recolección de datos, el análisis de resultados y la implementación de mejoras iterativas para asegurar que el sistema cumpla con los estándares de calidad y expectativas del cliente.

Tabla 3.4: Épicas del proyecto

ID	Descripción
RJDC-35	Documentar el proceso de sicronización
RJDC-36	Documentar código fuente del extractor
RJDC-37	Ajustar extractor para métricas
RJDC-38	Toma de métricas del extractor
RJDC-39	Diseño del estado futuro del sistema (TO BE)
RJDC-40	Atributos de calidad del sistema
RJDC-41	Prueba de concepto
RJDC-42	Elaborar plan de transición
RJDC-43	Desarrollo microservicio extractor-api
RJDC-44	Desarrollo de publisher
RJDC-45	Desarrollo de subscriber
RJDC-46	Desarrollo de extractor-cheker
RJDC-47	Capacitación equipo
RJDC-48	Manual de operación
RJDC-49	Manual de despliegue
RJDC-50	Levantar métricas del sistema diseñado
RJDC-51	Evaluación de requerimientos de calidad
RJDC-52	Implementar observabilidad

Tabla 3.5: Historias de usuario

3.3. Requerimientos

El sistema debe ser capaz de actualizar la información de los procesos judiciales registrados por los clientes. Estos procesos deben actualizarse consultando la información disponible en el aplicativo ramajudicial. Esta información debe extraerse y aquellas actuaciones que no hayan sido registradas deberán registrarse en la base de datos de redjudicial para que sean visibles por los clientes. Diariamente, el sistema deberá notificar a través de email a los clientes todos los movimientos que se encontraron durante el día.
El requerimiento de redjudicial se basa en la automatización de la extracción de datos de una página web, por lo cual a continuación se detallan los pasos que se deben seguir para obtener la información:

1. El usuario debe ingresar a la página `https://consultaprocesos.ramajudicial.gov.co/Procesos/NumeroRadicacion`, en esta página debe seleccionar si desea una búsqueda rápida (Datos recientes) o una búsqueda completa e ingresar el número del radicado del proceso a consultar

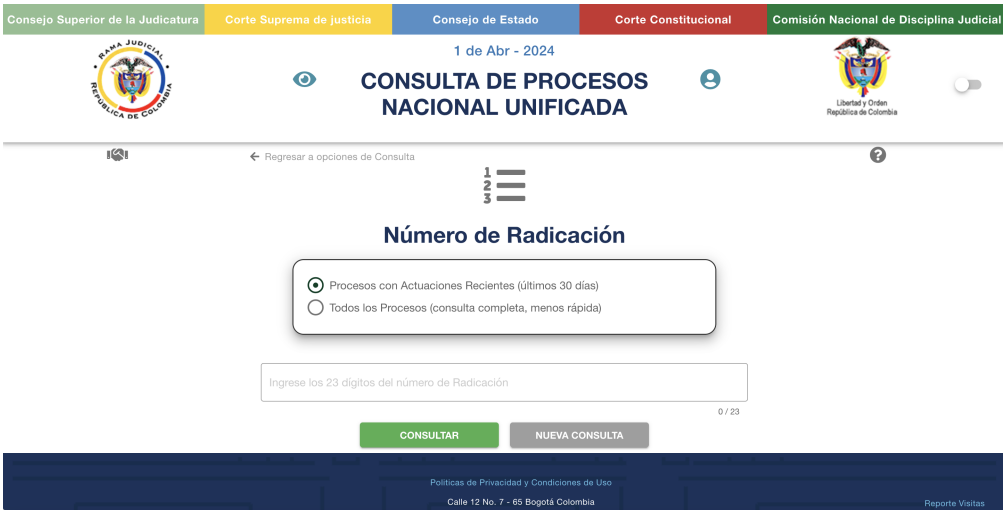

Figura 3.1: Pantalla de búsqueda rama judicial

2. Después de seleccionar el modo de búsqueda y presionar el botón "Consultar" el sistema le mostrará al usuario los resultados encontrados para ese proceso, el usuario podrá seleccionar el proceso.

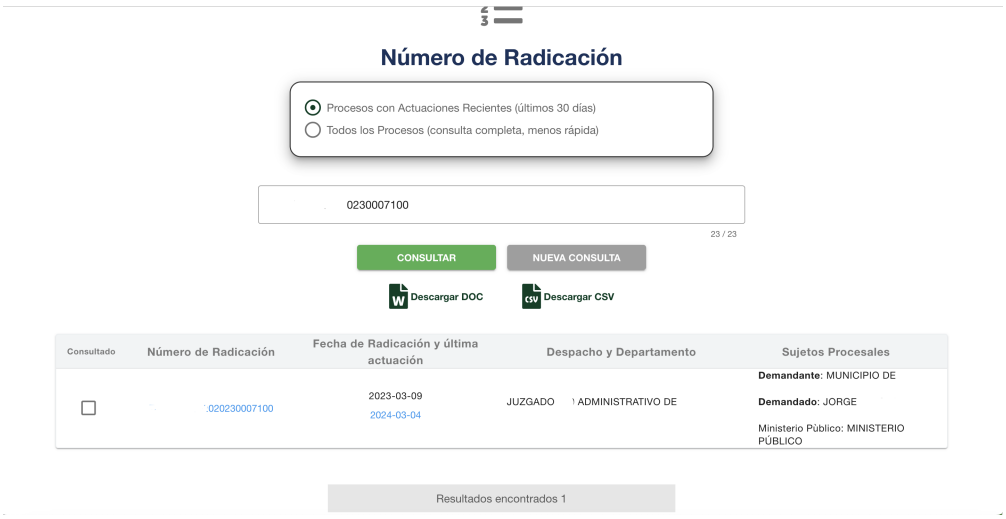

Figura 3.2: Pantalla de resultados rama judicial

En caso de que se haya seleccionado la búsqueda rápida y el proceso judicial no cuente con información reciente (Hasta 30 días), el sistema le informará al usuario que no hay resultados

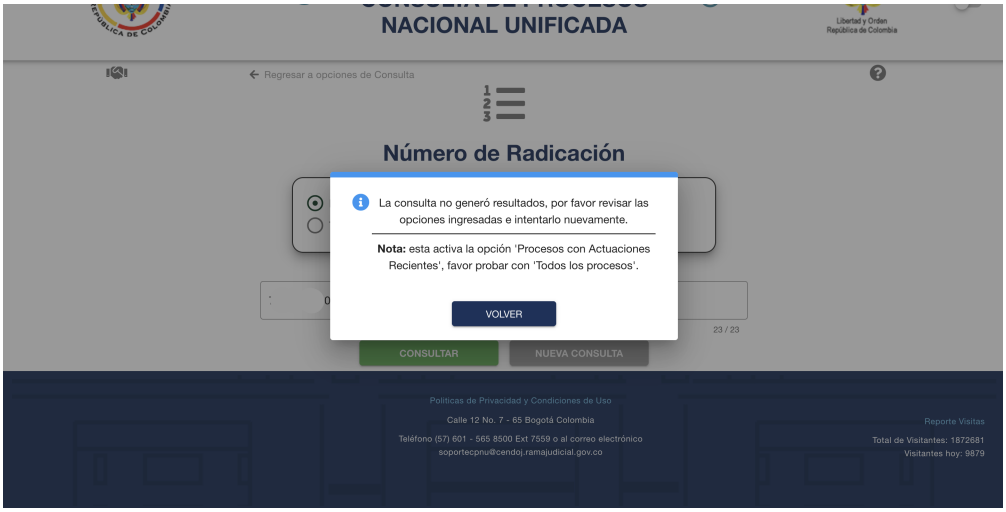

Figura 3.3: Pantalla informativa sin resultados Rama judicial

3. Una vez seleccionado el proceso, el sistema le mostrará al usuario la información del proceso a través de pestañas, para poder obtener las actuaciones deberá dar click sobre la pestaña "Actuaciones"

Figura 3.4: Pantalla de detalle del proceso judicial en Rama Judicial

En la pestaña de actuaciones están disponibles todas las actuaciones del proceso, estas serán las actualizaciones que se deben registrar.

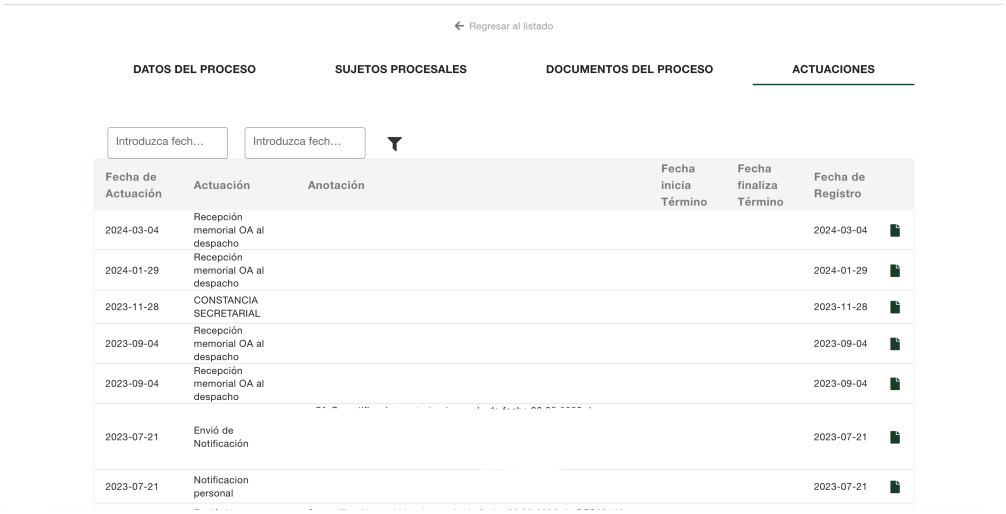

Figura 3.5: Detale de actuaciones del proceso rama judicial

3.3.1. Requerimientos funcionales

Fue crucial identificar y definir de manera clara los requerimientos funcionales en el proyecto, ya que representan las funcionalidades esenciales que el sistema debe cumplir para satisfacer las necesidades del usuario y los objetivos del negocio. Estos requerimientos proporcionan una guía clara para el desarrollo del sistema, permitiendo una comprensión precisa, una priorización adecuada, una validación efectiva y una adaptabilidad óptima a lo largo del ciclo de vida del proyecto.

Para levantar los requerimientos se empleó una estrategia integral que consideró tanto el análisis del sistema actual como la retroalimentación directa de los usuarios involucrados en el proceso. Esta combinación de fuentes permitió obtener una visión completa y detallada de las necesidades y desafíos existentes en el contexto de Redjudicial. Al estudiar el sistema actual, se identificaron las fortalezas y limitaciones técnicas, los puntos críticos de rendimiento y los aspectos que requieren mejoras. Por otro lado, la interacción directa con los usuarios proporcionó información valiosa sobre sus expectativas, requerimientos específicos y áreas de oportunidad para optimizar la experiencia y eficiencia del sistema. Integrando estas perspectivas, se pudo elaborar un conjunto de requerimientos alineados con las necesidades reales del proyecto Redjudicial.

Como resultado y a grandes rasgos, se identificó que el extractor tiene dos casos de uso importantes para redjudicial, no obstante estas funciones son complementarias entre sí, actualización de procesos judiciales y notificación de novedades. Estos requerimientos fueron revisados por el cliente y aprobados por él.

RQFN001 - Actualizar proceso judicial	
Descripción:	El extractor debe consultar la información de cada proceso judicial en rama judicial. Si el proceso tiene actuaciones recientes, estas deben registrarse en rama judicial para que puedan ser accedidas por los clientes. Para realizar esto, el sistema deberá ingresar a la página de ramajudicial. Si el proceso ha sido actualizado o consultado recientemente (menos de 30 días), el sistema seleccionará el modo de búsqueda rápida; de lo contrario, seleccionará la consulta completa. Luego, ingresará el número de radicado del proceso y presionará el botón de consulta, esperando a que la página muestre los resultados y accederá a él. Una vez que el sistema muestre la información del proceso judicial, deberá ingresar a la pestaña de actuaciones y extraer solo aquellas que no estén registradas en el sistema y registrarlas en redjudicial.
Usuario:	N/A Proceso automático
Fuente de datos:	Para este proceso el extractor debe consultar diferentes fuentes de ellas: ▪ **Redjudicial:** El sistema debe consultar la base de datos de Redjudicial para obtener la información de los procesos que serán consultados, además deberá realizar la actualización de datos en Redjudicial. ▪ **Rama judicial:** El sistema deberá consultar las actualizaciones del proceso en Rama juducial
Datos de entrada:	▪ Número de radicado del proceso ▪ Fecha en la que fue consultado por última vez
Resultado esperado:	El sistema debe registrar en Redjuducial las actuaciones encontradas en Ramajuducial, para lo cual deberá entregar la siguiente información por cada actuación: ▪ Número de radicado del proceso ▪ Fecha de actuación ▪ Fecha en la que inicia el término ▪ Fecha en la que finaliza el término ▪ Fecha de registro de la actuación

Tabla 3.6: Requerimiento Actualizar proceso judicial

RQFN001 - Actualizar proceso judicial	
Criterios de aceptación:	Para que este requerimiento se considere satisfecho debe cumplir los siguientes criterios: ▪ **CA001:** El sistema debe ser capaz de consultar los procesos pendiente por sincronizar priorizando aquellos que llevan más tiempo sin ser actualizados ▪ **CA002:** El sistema debe consultar en Ramajudicial si un proceso judicial tiene nuevas actuaciones, en caso de que las tenga deberá descargarlas, de lo contrario, deberá finalizar el proceso ▪ **CA003:** Al consultar las actuaciones de un proceso judicial en Ramajudicial el sistema deberá registrarlas en la base de datos y actualizar el proceso con la fecha de última actualización. En caso de que no tenga actuaciones solo deberá actualizar la fecha de actualización.

Tabla 3.7: Requerimiento Actualizar proceso judicial - continuación

RQFN002 - Notificación clientes	
Descripción:	Una vez al día, en una hora definida por el negocio, el sistema debe notificar a los clientes a través de email las actuaciones que se hayan encontrado para sus procesos.
Usuario:	N/A Proceso automático
Fuente de datos:	Para este proceso el extractor debe consultar diferentes fuentes de ellas: ▪ **Redjudicial:** Actuaciones encontradas en el día. ▪ **Email:** Servidor de correo a través del cual se enviará la notificación
Datos de entrada:	▪ Email del cliente ▪ Actuaciones encontradas por proceso
Resultado esperado:	El sistema debe enviar un email al cliente
Criterios de aceptación:	Para que este requerimiento se considere satisfecho debe cumplir los siguientes criterios: ▪ **AC004:** El sistema debe consultar las actuaciones que se hayan encontrado y que aun no hayan sido notificadas a los clientes. ▪ **AC005:** El sistema debe enviar un correo (Utilizando el API del proveedor de correos) con el listado de actuaciones encontradas relacionadas a su respectivo proceso judicial.

Tabla 3.8: Requerimiento Notificación clientes

3.3.2. Atributos de Calidad

Los requerimientos no funcionales son caracteristicas o calificaciones de las funcionalidades del sistema o del sistema en general (Bass et al., 2012), sin ser una función específicamente indican en qué condiciones deben realizarse las tareas del extractor. Estos requerimientos abordan aspectos como la usabilidad, rendimiento, seguridad, fiabilidad, mantenibilidad, compatibilidad, entre otros. Son igualmente importantes que los requerimientos funcionales, ya que contribuyen a la efectividad del sistema, su capacidad para cumplir con los objetivos del negocio y la experiencia del usuario.
La definición de requerimientos no funcionales es fundamental para el negocio y su promesa de valor, ya que a través de estos se logra el cumplimiento de la misma. La fiabilidad es especialmente importante para el negocio, porque sobre este atributo recae su promesa de valor, ya que Redjudicial debe garantizar que los procesos judiciales de sus clientes se actualizan oportunamente. Para

identificar estos requerimientos, se hicieron sesiones con los interesados.

- **Idoneidad funcional:** La idoneidad funcional es uno de los atributos de calidad que se enfoca en la capacidad del sistema para proporcionar las funciones necesarias que cumplen con los requisitos del usuario y del negocio. La idoneidad funcional es crucial, ya que el sistema debe gestionar de manera eficiente los procesos judiciales y satisfacer las necesidades específicas de los usuarios, tales como abogados y personal de soporte. La aplicación de este concepto en Redjudicial implica lo siguiente:

  - **Actualización de procesos judiciales:** El sistema debe ser capaz de realizar la vigilancia de los procesos judiciales, encontrando y registrando actuaciones.
  - **Notificación de movimientos:** El sistema debe ser capaz de notificar a los clientes en determinadas horas los movimientos encontrados.

  En la tabla 3.9 se definen las métricas con las cuales se medirá la idoneidad funcional del sistema.

Métricas de Idoneidad funcional	
Puntualidad de actualización:	Mide la puntualidad con la que se registran las actuaciones. Para cada actualización la puntualidad se mide como el número de días entre la fecha de la actuación y la fecha en la que se regista. Para el sistema en general la puntualidad se calculará como la puntualidad promedio.
Puntualidad de notificación:	Mide la puntualidad con la que las actuaciones encontradas se notifican a los clientes, partiendo de que todas las actuaciones deben ser notificadas en grupo a cada cliente en horarios específicos (cortes), por lo cual se medirá con la cantidad de cortes de notificación que tardó en ser notificada al cliente.
Vigilancia:	Para asegurar que los procesos estén actualizados a la fecha es necesario que el sistema revise varias veces en el mismo día los procesos, al menos aquellos que suelen tener más movimientos, esta métrica se calcula como el promedio de intentos exitosos por proceso judicial.

Tabla 3.9: Métricas de idoneidad funcional

- **Fiabilidad:** El sistema debe ser confiable, lo que significa que debe estar disponible y ser tolerante a los fallos que puedan presentarse, especialmente aquellos cuya fuente es Rama-Judicial, ya que esta es la principal fuente de información del sistema. Para el negocio, este atributo de calidad es de vital importancia, ya que Redjudicial debe garantizar a sus clientes que los procesos judiciales se actualizan correctamente todos los días. Por ello, el sistema debe asegurar que los procesos se revisen adecuadamente, incluso cuando no haya datos nuevos que

registrar, en la tabla 3.10 se definen las métricas a través de las cuales se medirá la fiabilidad del sistema.

Métricas de Fiabilidad	
Tasa de éxito:	Mide la proporción de intento de sincronización exitosas (Se considera exitosa siempre que se finalice satisfactoriamente independientemente de que encuentra actuaciones o no.) respecto al total de intentos de sincronización. Se calcula dividiendo el número de extracciones exitosas entre el número total de intentos de extracción.
Tasa de error:	Mide la proporción de errores en la sincronización respecto al total de intentos. Se calcula dividiendo el número de errores en las extracciones entre el número total de intentos de sincronización.
Tasa de reintentos	Mide la cantidad de intentos requeridos en promedio para una sincronización exitosa, se calcula dividiendo la cantidad total de intentos entre el número de procesos sincronizados satisfactoriamente

Tabla 3.10: Métricas de fiabilidad

- **Rendimiento:** El extractor debe ser capaz de consultar un gran volumen de procesos judiciales al día, consultar en rama judicial si estos procesos han tenido movimientos y registrarlos en SistemaJudicial. Dado el gran volumen de procesos y la intención de Redjudicial de seguir creciendo, el sistema debe poder procesar todos los procesos diariamente.

Métricas de Rendimiento	
Tiempo de sincronización por proceso:	El tiempo que toma la actualización de un proceso hasta que la información es visible en el aplicativo SistemaJudicial
Throughput:	Cantidad de procesos que el sistema puede actualizar de manera efectiva por hora.
Uso de memoria RAM:	Cantidad de Ram requerida por el sistema para funcinar, se deben tener en cuenta los picos
Uso de CPU:	Porcentaje de CPU utilizado por el sistema, se deben tener en cuenta los picos.

Tabla 3.11: Métricas de rendimiento

- **Escalabilidad:** Debe ser capaz de escalar para manejar un mayor volumen de datos o usuarios en el futuro sin comprometer el rendimiento. Con el tiempo, la cantidad de información relacionada con procesos judiciales puede aumentar significativamente. Una arquitectura escalable permite manejar grandes volúmenes de datos de manera eficiente y sin comprometer

el rendimiento. Esto es fundamental para los planes de crecimiento del negocio, ya que no sería eficiente tener que rediseñar o implementar nuevo código para aumentar la flexibilidad del sistema, además de permitirle optimizar el uso de recursos, ya que puede adaptarse a las necesidades del sistema en cada momento. Esto puede traducirse en ahorros significativos en costos operativos y de mantenimiento a medida que el sistema crece.

- **Seguridad:** La seguridad juega un papel crítico en la protección de los activos de información y la garantía de la confidencialidad, integridad y disponibilidad de los sistemas (Bass et al., 2012). En el contexto de Redjudicial, la seguridad es un componente fundamental debido a la naturaleza sensible de la información manejada, como datos judiciales y procesos legales. Esto requiere medidas de seguridad robustas para prevenir accesos no autorizados, proteger la información contra modificaciones indebidas y asegurar que los servicios estén disponibles cuando se requieran.

  La seguridad abarca diversos aspectos, como la autenticación de usuarios para garantizar que solo personal autorizado acceda al sistema, el control de acceso para limitar los privilegios según roles y responsabilidades, la encriptación de datos tanto en reposo como en tránsito para proteger la confidencialidad, y la implementación de mecanismos de auditoría y monitoreo para detectar y responder a posibles amenazas o incidentes de seguridad de manera oportuna.

  La confianza es un valor fundamental, por lo cual Redjuducial debe cuidar de la información de sus sistemas ya que, además, en este se sustenta su operación.

- **Mantenibilidad:** El extractor de Redjudicial debe ser fácil de mantener, de tal manera que la puesta en marcha, el soporte y las modificaciones no tengan gran complejidad. A nivel de código debe ser fácilmente modificable ya que la página Ramajudicial puede cambiar en cualquier momento y se requerirían cambios en el sistema de scrapping.

- **Interoperabilidad:** Dado que el extractor estará interactuando con Ramajudicial y sistemaJudicial, es importante que el extractor pueda intercambiar datos con otros sistemas judiciales, gubernamentales o de terceros. La interoperabilidad garantiza que estos sistemas puedan interactuar de manera efectiva y sin problemas, y debe poder acceder a datos compartidos de manera segura y eficiente. lo cual es especialmente importante en el contexto judicial, donde la disponibilidad de información precisa y actualizada es fundamental para la toma de decisiones y la administración de casos. Una buena interoperabilidad evitará que el sistema esté fragmentado, donde la información relevante se encuentra dispersa en diferentes sistemas o formatos. Al asegurar la interoperabilidad, Redjudicial puede evitar esta fragmentación y garantizar la coherencia y la integridad de sus datos.

3.3.3. Escenarios de calidad

Una vez definidos los atributos de calidad, se definieron los escenarios de calidad en los que se pondrán a prueba estos atributos, estos escenarios. (Bass et al., 2021) establecen una forma en la cual se pueden describir los escenarios de calidad del sistema desde diferentes perspectivas

(desempeño o experiencia de usuario). En la tabla 3.12 se identifican los atributos con los que se pueden especificar un escenario de calidad.

Nombre del escenario	
Atributo de Calidad:	Atributo de Calidad que será evaluado
Fuente del estímulo:	Los estímulos tienen una fuente, vienen de algún lado o ente (usuario, sistema, otro actor)
Estímulo:	Es un evento que llega al sistema, puede ser un evento o una operación del uuario
Artefacto:	El estímulo llega a un objetivo que puede ser el sistema en sí o un componente del mismo o una colección de sistemas.
Entorno:	Conjunto de circunstancias en las cuales el escenario toma lugar, Usualmente se refiere a un estado en tiempo de ejecución.
Respuesta:	La actividad o acción que se debe desencadenar como resultado del evento
Medición de la respuesta:	Cuando la respuesta ocurre debe ser medible, de esta manera determinamos si el requeirmiento de calidad se logra con la implementación.

Tabla 3.12: Estructura de definición de un escenario de calidad

Basado en este formato se han establecido los requerimientos de calidad del sistema de Redjudicial, estos escenarios se han generado por atributo de calidad.

Adecuación Funcional - Escenario de Calidad RJQS001	
Atributo de Calidad:	Adecuación Funcional
Fuente:	Sistema Redjudicial
Estímulo:	Solicitud de actualización de un proceso judicial
Descripción:	El sistema debe ser capaz de actualizar un proceso judicial. Este escenario tiene un enfoque funcional, el sistema debe consultar la información e insertarla correctamente en la base de datos.
Artefacto:	Extractor
Entorno:	Operación normal
Respuesta:	El sistema actualiza el proceso judicial en la base de datos
Medición de la respuesta:	Procesos judiciales actualizados correctamente

Tabla 3.13: Escenario de calidad RJQS001: Adecuación Funcional Sincronización de procesos judiciales

Rendimiento - Escenario de Calidad RJQS002	
Atributo de Calidad:	Rendimiento
Fuente:	Sistema Redjudicial
Estímulo:	Aumenta el número de procesos judiciales pendientes
Descripción:	El extractor debe ser capaz de aprovechar los recursos hardware del sistema para incrementar la capacidad de procesamiento.
Artefacto:	Extractor
Entorno:	Alta carga de trabajo
Respuesta:	El sistema procesa las solicitudes de actualización de manera concurrente. Cada nodo del sistema debe ser capaz de procesar mínimo 4 procesos por minuto.
Medición de la respuesta:	Procesos sincronizados por minuto por nodo

Tabla 3.14: Escenario de calidad RJQS002: Rendimiento Sincronización de procesos judiciales

Tolerancia a Fallos - Escenario de Calidad RJQS003	
Atributo de Calidad:	Tolerancia a Fallos
Fuente:	Ramajudicial
Estímulo:	Incremento en la latencia de Ramajudicial
Descripción:	Cuando hay alta carga de trabajo el rendimiento de Ramajudicial se degrada, por lo cual hay tiempos de espera más altos al realizar el scraping, esto puede hacer que el extractor falle o que el extractor se detenga si se queda esperando.
Artefacto:	Extractor
Entorno:	Alta carga de trabajo
Respuesta:	El sistema reintenta la conexión con el componente de Rama Judicial utilizando una estrategia de polling hasta que se cumpla 1 minuto
Medición de la respuesta:	Porcentaje de errores de timeout. Duración sincronización promedio. El porcentaje de solicitudes fallidas por timeout debe ser inferior al 40 %.

Tabla 3.15: Escenario de calidad RJQS003: Tolerancia a Fallos Sincronización de procesos judiciales

Tolerancia a Fallos - Escenario de Calidad RJQS004	
Atributo de Calidad:	Tolerancia a Fallos
Fuente:	Ramajudicial
Estímulo:	Ramajudicial no está disponible
Descripción:	Cuando hay alta carga de trabajo el rendimiento de Ramajudicial puede dejar de funcionar progresivamente, por lo cual no es capaz de procesar las solicitudes del extractor y esto genera errores. Cuando Ramajudicial falla el sistema debe reducir la carga para que pueda recuperarse y vuelva a estar disponible.
Artefacto:	Extractor
Entorno:	Indisponibilidad Ramajudicial
Respuesta:	El sistema reduce el número de solicitudes que hace a Ramajudicial hasta que responda nuevamente
Medición de la respuesta:	Solicitudes por minuto se debe reducir el número cada que se presentan errores de disponibilidad de Ramajudicial.

Tabla 3.16: Escenario de calidad RJQS004: Tolerancia a Fallos Sincronización de procesos judiciales

Tolerancia a Fallos - Escenario de Calidad RJQS005	
Atributo de Calidad:	Tolerancia a Fallos
Fuente:	Sistema Redjudicial
Estímulo:	Fallo en la actualización de un proceso judicial
Descripción:	Algunos procesos judiciales tienen datos incompletos (que son requeridos) o se genera un error inesperado en el sistema. El sistema debe manejar y reportar estos errores para poder seguir funcionando. El sistema deberá generar alertas cuando incrementen este tipo de errores.
Artefacto:	Extractor
Entorno:	Fallo no determinado
Respuesta:	El sistema registra el fallo, reintenta la actualización.
Medición de la respuesta:	Errores capturados, Cantidad de excepciones por tipo

Tabla 3.17: Escenario de calidad RJQS005: Tolerancia a Fallos Sincronización de procesos judiciales

Tolerancia a Fallos - Escenario de Calidad RJQS006	
Atributo de Calidad:	Tolerancia a Fallos
Fuente:	Sistema Redjudicial
Estímulo:	El sistema vuelve a funcionar después de una falla.
Descripción:	Después de una falla el sistema debe retomar la sincronización en el punto donde estaba
Artefacto:	Extractor
Entorno:	Recuperación del sistema
Respuesta:	El sistema debe continuar la sincronización en el punto donde estaba antes del fallo.
Medición de la respuesta:	Procesos judiciales sin actualizar, el número debe ser menor en cada iteración del proceso. ▪ **RTO (Recovery Time Objective):**Dado que Redjudicial opera en un entorno crítico para sus clientes (procesos judiciales sensibles y con plazos legales estrictos), cualquier interrupción prolongada podría impactar severamente la confianza de los clientes y la calidad del servicio. Un RTO de 8 horas asegura que el sistema esté disponible rápidamente tras una interrupción.

Tabla 3.18: Escenario de calidad RJQS006: Tolerancia a Fallos Sincronización de procesos judiciales

Interoperabilidad - Escenario de Calidad RJQS007	
Atributo de Calidad:	Interoperabilidad
Fuente:	Sistema Redjudicial
Estímulo:	Consulta a Ramajudicial
Descripción:	El sistema debe ser capaz de comunicarse con Ramajudicial.
Artefacto:	Extractor
Entorno:	Operación normal
Respuesta:	El sistema interactúa con los componentes web de Ramajudicial
Medición de la respuesta:	Sincronizaciones exitosas

Tabla 3.19: Escenario de calidad RJQS007: Interoperabilidad Sincronización de procesos judiciales

Escalabilidad - Escenario de Calidad RJQS008	
Atributo de Calidad:	Escalabilidad
Fuente:	Sistema Redjudicial
Estímulo:	Aumento en el número de solicitudes de actualización
Descripción:	Cuando aumenta la carga del sistema (número de procesos judiciales) el sistema debe permitir agregar más instancias del mismo (escalar) para incrementar su capacidad.
Artefacto:	Extractor
Entorno:	Alta carga de trabajo
Respuesta:	El sistema permite configurar más nodos y distribuye la carga de trabajo entre los nuevos nodos. Inicialmente estos nodos deberán ser configurados manualmente, basados en el rendimiento evidenciado en el dashboard del sistema. Esta función se automatizará a través de la implementación de devops (no considerada aun)
Medición de la respuesta:	El sistema permite tener más de una instancia de un componente

Tabla 3.20: Escenario de calidad RJQS008: Escalabilidad Sincronización de procesos judiciales

Adecuación Funcional - Escenario de Calidad RJQS009	
Atributo de Calidad:	Adecuación Funcional
Fuente:	Sistema Redjudicial
Estímulo:	Notificación de actuaciones encontradas a horas específicas
Descripción:	Cuando llega una de las horas señaladas para enviar notificaciones el sistema debe enviar un correo a cada cliente con las actuaciones que se han encontrado hasta el momento.
Artefacto:	Extractor
Entorno:	Operación normal
Respuesta:	El sistema envía notificaciones de actuaciones encontradas a los clientes a horas específicas
Medición de la respuesta:	Las notificaciones deben ser enviadas puntualmente en la hora especificada sin fallos. Las actuaciones deben ser notificadas el mismo día que se registraron o en la mañana del día siguiente.

Tabla 3.21: Escenario de calidad RJQS009: Adecuación Funcional Notificación de actuaciones de procesos judiciales

Interoperabilidad - Escenario de Calidad RJQS010	
Atributo de Calidad:	Interoperabilidad
Fuente:	Sistema Redjudicial
Estímulo:	Enviar notificación a clientes
Descripción:	El sistema debe integrarse con el proveedor de notificaciones.
Artefacto:	Extractor
Entorno:	Operación normal
Respuesta:	El sistema se conecta con el proveedor de email para enviar las notificaciones
Medición de la respuesta:	Las notificaciones deben ser enviadas puntualmente en la hora especificada sin fallos

Tabla 3.22: Escenario de calidad RJQS010: Interoperabilidad Notificación de actuaciones de procesos judiciales

Tolerancia a Fallos - Escenario de Calidad RJQS011	
Atributo de Calidad:	Tolerancia a Fallos
Fuente:	Sistema Redjudicial
Estímulo:	Enviar notificación a clientes
Descripción:	Al enviar notificaciones el sistema debe manejar los errores que se presenten y reintentar en caso de fallos. El sistema siempre debe reportar los errores que se presenten.
Artefacto:	Extractor
Entorno:	Error en el sistema
Respuesta:	El sistema reporta el error
Medición de la respuesta:	Las notificaciones deben ser enviadas puntualmente en la hora especificada sin fallos

Tabla 3.23: Escenario de calidad RJQS011: Tolerancia a Fallos Notificación de actuaciones de procesos judiciales

3.3.4. Restricciones de Arquitectura.

Una restricción de arquitectura es una decisión o directiva que limita las opciones de diseño de un sistema de software. Estas restricciones pueden ser impuestas por el entorno en el que opera el sistema, por requisitos de rendimiento, seguridad, interoperabilidad, o por decisiones estratégicas de la organización. Las restricciones de arquitectura son importantes porque definen los límites dentro de los cuales se debe desarrollar la arquitectura del sistema, garantizando así que cumpla con los objetivos y requerimientos establecidos. (Bass et al., 2012)

Restricciones de arquitectura	
AR001	El sistema debe ser diseñado dando prioridad a la infraestructura On-Premise pero con posibilidad de una futura migración en la nube.
AR002	Debe usarse C# como lenguaje de programación.
AR003	Solo los empleados de Redjudicial pueden acceder al extractor.
AR004	El sistema debe permitir la autenticación utilizando la lista de usuarios que actualmente existen para Redjudicial.
AR005	Para mantener la simplicidad del sistema, no se implementarán procesos DevOps
AR006	Para implementaciones Cloud se debe priorizar el uso de las máquinas locales de Redjudicial para el proceso de scraping

Tabla 3.24: Restricciones del sistema

3.4. Estado actual del sistema (As-Is)

En el contexto de la evaluación y mejora continua de sistemas de información, el análisis del estado actual (As-Is) del proyecto Redjudicial se erige como un pilar fundamental. Este proceso parte de una premisa esencial: comprender exhaustivamente cómo opera el sistema en su configuración actual. Esta comprensión se logra mediante la identificación de los procesos, componentes y flujos de información que componen el entramado del proyecto.

El análisis As-Is se nutre de los requerimientos del sistema y los objetivos del negocio, establecidos como pilares fundamentales en el diseño y evolución de Redjudicial. Su propósito es esclarecer y documentar en detalle qué es lo que hace el sistema en su estado presente. Esta labor va más allá de una mera descripción; implica una evaluación profunda de cada función, interacción y procedimiento dentro del sistema.

El objetivo principal de este análisis es identificar de manera precisa los puntos de falla, ineficiencias y limitaciones que puedan estar afectando el desempeño y la eficacia de Redjudicial en su estado actual. Al hacerlo, se abre la puerta a oportunidades de mejora significativas que apuntan a optimizar la operatividad, la usabilidad y la satisfacción de los usuarios finales.

3.4.1. Recopilación de datos

La recopilación de datos del estado actual del sistema en el proyecto Redjudicial constituye una fase crítica dentro del proceso de evaluación y mejora continua. En esta etapa, se llevó a cabo una revisión exhaustiva de los sistemas existentes, el código subyacente y el funcionamiento de cada componente que conforma el entorno de Redjudicial. El propósito principal fue obtener métricas precisas y representativas que permitieran evaluar la calidad del sistema en su configuración actual. Durante esta revisión, se analizaron detalladamente cada uno de los procesos, flujos de información y funcionalidades del sistema. Se buscó comprender cómo interactúan los diferentes componentes y cómo se comportan en conjunto para cumplir con los objetivos y requerimientos establecidos. Esta

comprensión profunda del sistema permitió identificar áreas de mejora, puntos críticos y posibles puntos de falla que necesitan atención.
Es importante destacar que algunas métricas de calidad del sistema no estaban disponibles de manera directa, por lo cual se hicieron ajustes que permitieran recopilar aquellas de gran relevancia para el funcionamiento del sistema.
Se reconoció la importancia de estas métricas para la toma de decisiones informadas y la priorización de acciones de mejora. Por ello, se hizo hincapié en obtener datos confiables y relevantes que respalden el análisis y la evaluación del estado actual del sistema en el proyecto Redjudicial. Esta labor de recopilación de datos es fundamental para fundamentar las decisiones de diseño, optimización y evolución del sistema, asegurando su alineación con los estándares de calidad y las expectativas de los usuarios finales y stakeholders clave.
Estas pruebas se hicieron en un ambiente aislado para garantizar que no se afectara la operación de Redjuducial y que los movimientos de la operación no afectaran los resultados. Para ello se usó un servidor independiente en el cual se instaló el extractor. Para este proceso se agregaron logs específicos en el extractor que permitieron identificar cuándo un proceso inicia, cuándo finaliza y detalles importantes del proceso como excepciones y el tipo de excepciones; adicionalmente, se modificó el extractor para, a través de feature-flags, activar solo una parte de los clientes, de tal manera que el extractor de la prueba solo consultaría procesos judiciales de 3 clientes y el extractor real ejecutaría los demás.

Hardware	
Sistema operativo	Windows server 2022
Memoria RAM	8GB
CPU	4
Framework	Netframwork 6.0

Tabla 3.25: Hardware utilizado para la obtención de métricas.

3.4.2. Diseño de Alto nivel

El diseño de alto nivel permitió comprender cómo está estructurado el sistema actual, de tal manera que se puedan identificar sus componentes, dependencias y relaciones.

3.4.2.1. Diagrama de Contexto

En la figura 3.6 se muestra el diagrama de contexto, el cual proporciona una visión global del sistema y sus interacciones externas. En este diagrama se definieron los componentes internos y externos del sistema, se identifican las interacciones entre ellos, proporciona un mapa claro de las interfaces y dependencias externas que deben ser consideradas durante el diseño y la implementación del sistema.

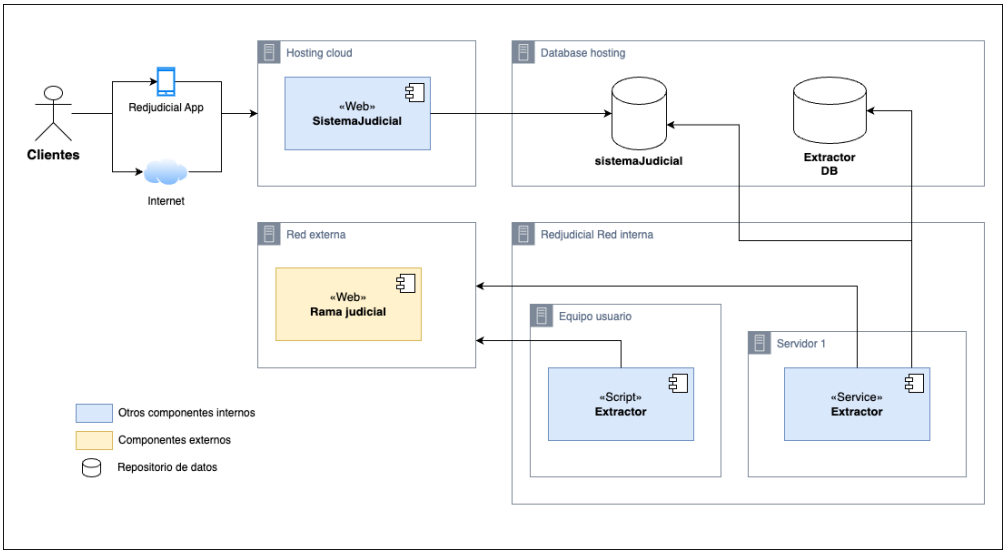

Figura 3.6: Vista de Contexto estado actual

- **Bases de datos:** Redjudicial cuenta con dos bases de datos MySql en un proveedor en la nube. Las bases de datos de Redjudicial en la nube se encuentran en el mismo motor, por lo cual todos los componentes están acoplados a ella.
- **Extractor:** Servicio desarrollado en C# netcore 6.0 que a través de la librería selenium y chromedriver se encarga de extraer la información desde la página web rama judicial e insertarla en la base de datos para que pueda ser accedida por los demás componentes.
- **Script Extractor:** Es un script desarrollado en python que se parametriza con un listado de procesos en excel y a través de selenium y chromedriver hace scrapping a la página de ramajudicial para extraer las actuaciones de los procesos judiciales, posteriormente registra estas actuaciones en el excel y luego son gestionadas por el personal de redjudicial quien la envía a través de correo electrónico a sus clientes. Este script es de uso temporal ya que no está integrado con el sistema y requiere ser revisado por el usuario que lo ejecuta.

3.4.2.2. Diagrama de despliegue

El diagrama de despliegue en la figura 3.7 es una representación del estado actual de la infraestructura tecnológica del sistema, mostrando cómo están distribuidos sus componentes, lo que permitió comprender su funcionamiento, identificar problemas o limitaciones, y servir de base para planificar mejoras hacia una arquitectura más eficiente (To-Be). Fue esencial para la toma de decisiones técnicas y para alinear al equipo en torno al estado actual del despliegue y los cambios necesarios.

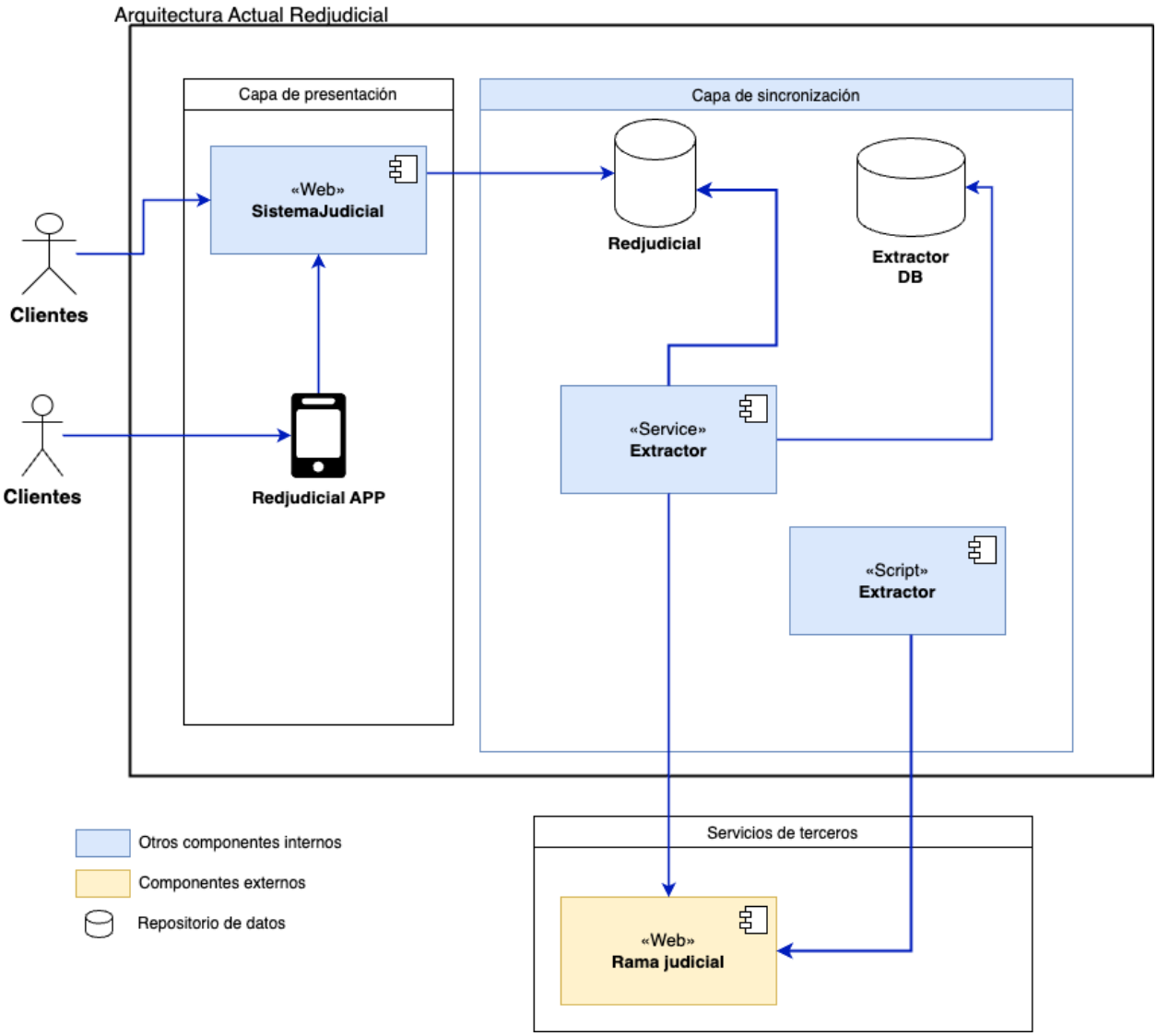

Figura 3.7: Vista de Despliegue estado actual

Actualmente Redjudicial cuenta con aplicativos desplegados en servidores en la nube, pero el proceso de extracción se realiza en un servidor local. La empresa cuenta con servidores que podrían usarse en el proceso de sincronización, pero no están en uso y el sistema actual no permite que se usen.

3.4.2.3. Diagrama de concurrencia

En la figura 3.8 se puede visualizar cómo se gestionan y coordinan las tareas concurrentes en el sistema en su estado inicial. Este diagrama permite identificar los procesos que se ejecutaban

simultáneamente, cómo interactúan entre sí y cómo se manejan los recursos compartidos, lo que ayuda a detectar posibles problemas de rendimiento, bloqueos o condiciones de carrera. Además, es una herramienta clave para optimizar la eficiencia del sistema y planificar mejoras en la gestión de la concurrencia en el diseño del sistema (To-Be).

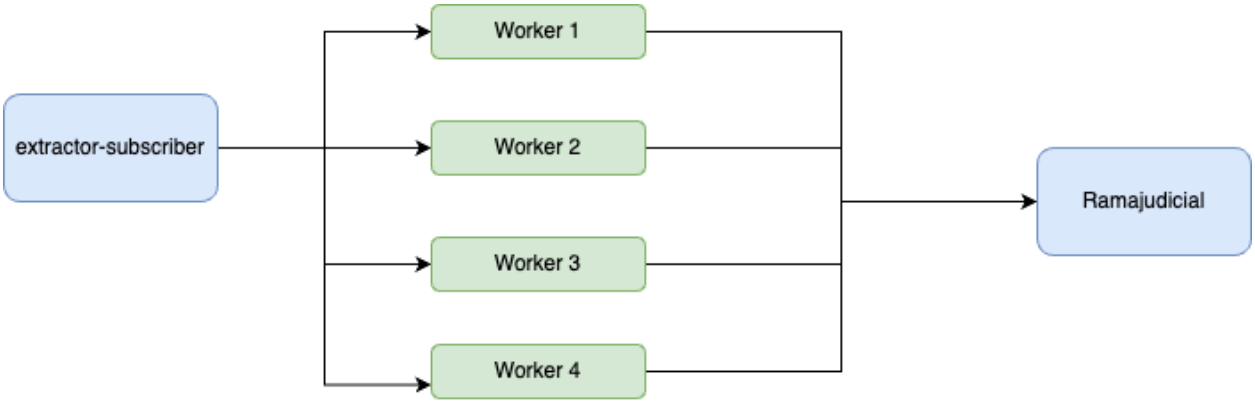

Figura 3.8: Vista de Concurrencia estado actual

Este modelo permite aumentar la capacidad de procesamiento en paralelo del sistema, pero el hecho de que esté acoplado hace que cualquiera de los errores que se puedan presentar, ya sea por scraping, uso de recursos compartidos o datos, afecten el funcionamiento de todo el sistema. Además, no es completamente eficiente, ya que el número de procesos en paralelo está limitado por recursos hardware.

3.4.3. Métricas de calidad As-Is

Los logs del sistema proporcionaron una visión de cómo se comporta el sistema en tiempo real, qué operaciones realiza, cuánto tiempo tardan en completarse, entre otros aspectos relevantes. Al analizarlos, se pudo identificar el rendimiento, la disponibilidad, la fiabilidad y otros aspectos críticos del sistema. Por ejemplo, se revisó la tasa de éxito en las operaciones, la frecuencia de errores, el tiempo de respuesta del sistema, entre otros indicadores clave.
Esta información fue fundamental para determinar qué aspectos del sistema requerían mejoras y optimizaciones. Se identificaron patrones de comportamiento, se realizaron análisis de tendencias y se compararon las métricas con los requerimientos establecidos. De esta manera, se pudo tener una imagen clara del estado actual del sistema y tomar decisiones informadas para implementar mejoras que impactaran positivamente en la calidad y el rendimiento del sistema de Redjudicial.
En la tabla 3.26 se muestran las métricas de fiabilidad del sistema.

Fiabilidad del sistema		
Métrica	**Valor**	**Descripción**
Tasa de éxito:	88 %	El sistema es capaz de procesar con éxito el 88 % de las solicitudes que realiza.
Tasa de error:	12 %	El sistema falla en el 12 % de las solicitudes que realiza.
Reintentos	0.95	El sistema require 1.95 reintentos para realizar la actualización de los procesos judiciales.
Cobertura:	15.72 %	El sistema lograr cubrir el 15.72 % de la carga.
Frecuencia:	2	El sistema revisa cada proceso al menos 2 veces para asegurarse de que se obtengan todos los cambios. Esta revisión solo se realiza sobre el 15.72 % de la carga.
Precisión:	35 %	El sistema registra el 70 % de las actualizaciones encontradas a tiempo.

Tabla 3.26: Métricas de fiabilidad

En la tabla 3.27 se muestran las métricas de rendimiento del sistema al procesar la carga de trabajo establecida, el tiempo promedio por proceso es de 91 segundos y el sistema puede llegar a ejecutar hasta 4 procesos en simultáneo ya que la ejecución de procesos se hace en paralelo, con lo cual llega a procesar hasta 179 procesos por hora. La gran cantidad de tiempo que toma sincronizar un solo proceso judicial está ligada a los tiempos de espera establecidos para los componentes web dinámicos. Esta capacidad de procesamiento también se ve afectada por tiempos muertos que se generan como mecanismo para reducir la tasa de errores. Cuando se presenta una falla, el sistema reduce el número de procesos que ejecutará en paralelo por 3 minutos, de esta manera Ramajudicial tiene tiempo de recuperarse .

Rendimiento del sistema		
Métrica	**Valor**	**Descripción**
Tiempo de sincronización por proceso	91s	El sistema se tarda en promedio 91 segundos en consultar la información de un proceso y un máximo de 100 segundos.
Rendimiento	176.67	El sistema es capaz de actualizar en promedio 179 procesos por hora.
Uso de memoria RAM:	30 %	La ejecución del extractor genera un incremento del 30 % de uso de memoria ram del servidor.
Uso de CPU:	40	La ejecución del extractor genera un incremento del 40 % de uso de CPU del servidor,

Tabla 3.27: Métricas de Rendimiento

El sistema actualmente es escalable verticalmente; al agregar más recursos (procesador, principalmente) puede aumentar el número de procesos a sincronizar en paralelo. No obstante, no puede

escalar horizontalmente.

3.4.4. Errores del sistema

A través de los logs también se pudo identificar cuáles son los errores por los cuales un intento de sincronización no se lleva a cabo satisfactoriamente, en la tabla 3.28 se identifican los errores encontrados en la ejecución del extractor y la ocurrencia de los mismos, en su mayoría se debe a intentos de acceder a componentes web dentro de la página que no existen o no han cargado aún. En el caso de errores relacionados con el scrapping, la ocurrencia se incrementa con el número de solicitudes, ya que la causa de estos es el tiempo que toma a un componente web estar disponible. Este aumento en la latencia, causado por el incremento en el número de las solicitudes, es un comportamiento de Ramajudicial que se debe controlar para poder reducir la tasa de errores.

Errores del sistema			
Error	**Ocurrencia**	**Tipo**	**Descripción**
Errores no controlados	12	Código	Errores en la manipulación de componentes web o valores incorrectos o nulos
Errores de Chromedriver	11	Scrapping	Errores generados al cargar el web driver, ya sea por conexión o memoria disponible para iniciar el navegador o compatibilidad
Respuesta de componentes	256	Scrapping	Error al obtener un componente, este error se genera cuando se está intentando acceder a un componente que no existe dentro de la página, en el caso de Ramajudicial, al ser una página dinámica, el contenido se va cargando dependiendo de la interacción del usuario, si la página se demora en cargar, el driver no encontrará el componente y generará esta excepción.
Errores de Red	1	Red	Error de red, el sistema no logra conectarse a la red.

Tabla 3.28: Errores del sistema: Excepciones

3.4.5. Análisis del estado actual

El presente análisis se centra en la evaluación y diagnóstico del extractor de información utilizado en el proyecto Redjudicial. La evaluación se ha llevado a cabo con el objetivo de identificar las problemáticas existentes en el funcionamiento actual del extractor y determinar qué aspectos cumplen o no con los requerimientos de calidad establecidos por el cliente. Este proceso de evaluación es esencial para comprender las capacidades y limitaciones del sistema, así como para proponer mejoras que optimicen su rendimiento y eficiencia.

Durante la evaluación, se ha identificado una serie de requerimientos de calidad que el cliente considera fundamentales para el correcto funcionamiento del extractor. Estos requerimientos abarcan aspectos como la velocidad de procesamiento, la fiabilidad en la extracción de datos, la disponibilidad del sistema, entre otros. La comparación de estos requerimientos con el estado actual del extractor ha revelado discrepancias significativas que requieren atención y acción inmediata.
Es importante destacar que las problemáticas identificadas en el extractor tienen diversas causas, que van desde limitaciones técnicas hasta errores en el diseño o implementación del sistema. Por lo tanto, es necesario abordar cada problema de manera específica y adaptada a su origen, con el fin de garantizar soluciones efectivas y duraderas.
En las siguientes secciones de este análisis, se detallarán las problemáticas identificadas en el extractor, se analizarán sus causas subyacentes y se propondrán estrategias y soluciones para mejorar el desempeño y la calidad del sistema.

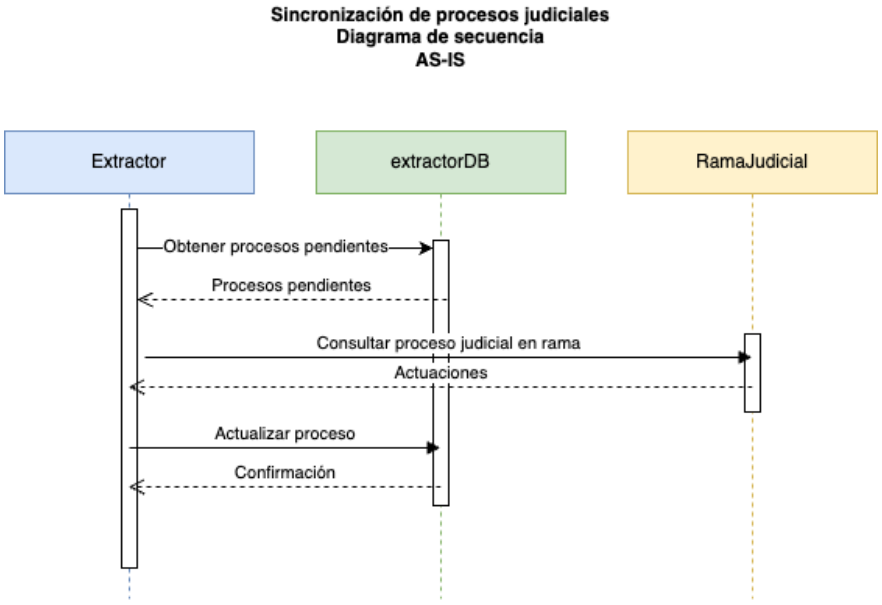

Figura 3.9: Diagrama de secuencia sincronización de datos

Problema	Análisis
El sistema no es capaz de sincronizar todos los procesos judiciales asignados.	Aunque el extractor fue capaz de hacer 2400 intentos, esos intentos solo abarcaron 1140 de ellos, en algunos casos se hicieron varios reintentos por proceso judicial. Actualmente el sistema accede a la base de datos y consulta los procesos judiciales y luego los itera de manera paralela, aunque esto aumenta el número de intentos, no es suficiente y además es costoso ya que solo se puede escalar con hardware
Distribución de la carga de trabajo	El sistema está consultando los procesos judiciales para sincronización, pero esta consulta no está organizando todos los datos de tal manera que todos los procesos sean sincronizados ya que algunos se reintentan aunque ya hayan sido sincronizados y para los demás no se realiza ningún intento.
Escalabilidad	Al escalar, el sistema puede aumentar la capacidad para procesar solicitudes. El extractor solo puede escalar verticalmente, lo cual limita la posibilidad de aumentar la carga de trabajo y se limita a la instancia en la que se ejecuta.
Observabilidad	Actualmente el sistema cuenta con logs en archivos planos, esto limita la visualización de los mismos, dificulta el análisis en tiempo real y aumenta el riesgo de perdida. Además de esto, el extractor ha dejado de funcionar ocasionalmente y Redjudicial no tiene una manera de identificarlo más que revisando el archivo de logs.
Gestión de tiempos de espera	El sistema toma demasiado tiempo en la consulta a Ramajudicial, esto se debe a que para prevenir fallos, el sistema cuenta con tiempos de espera de 30 segundos para dos componentes cuya carga se retrasa cuando la latencia se incrementa, este tiempo de espera se aplica para todas las búsquedas con alta y baja latencia.
Gestión de fallos	El único método que tiene el sistema para afrontar los fallos es detener la operación con tiempos de espera de 3 minutos, aunque la tasa de errores actualmente es baja, al incrementar la carga de trabajo esta se verá afectada y requerirá un método robusto para gestionarlos.

Tabla 3.29: Errores del sistema: Excepciones

3.5. Diseño del sistema (To Be)

Para llegar al estado ideal del sistema, el extractor Redjudicial se debe convertir en una plataforma tolerable a fallos, escalable y eficiente para la sincronización de procesos judiciales.
La arquitectura de Redjudicial se basa en principios fundamentales que garantizan su eficacia y capacidad para evolucionar junto con el negocio. La arquitectura ideal debe ser modular, adaptable y centrada en los atributos de calidad clave como el rendimiento, la confiabilidad y la seguridad. (Bass et al., 2012).
Garantizar modularidad es esencial para permitir que la arquitectura se pueda escalar y modificar sin la necesidad de afectar todo el sistema. Esto significa que cada componente debe tener una responsabilidad clara y ser independiente de otros, lo que facilita la evolución y el mantenimiento a largo plazo, permitiendo que la arquitectura responda de manera efectiva a los cambios en los requisitos del negocio y las tecnologías emergentes (Martin, 2017). Este principio se usa como base para garantizar los demás requerimientos de calidad del sistema, ya que esta capacidad es transversal para todo el sistema.
La fiabilidad es un aspecto crítico, ya que el sistema debe ser capaz de manejar fallas de manera efectiva y así estar disponible en todo momento. Esto incluye la implementación de mecanismos de tolerancia a fallos que permitan una rápida recuperación de los errores externos e internos, la gestión adecuada de excepciones y la observabilidad del sistema.
En términos de rendimiento, la arquitectura está diseñada para optimizar el uso de recursos y proporcionar tiempos de respuesta más cortos. Esto implica una cuidadosa planificación de la distribución de la carga de trabajo, el uso eficiente de recursos hardware, y la implementación de técnicas de optimización de código.

3.5.1. Decisiones de arquitectura

Las decisiones de arquitectura son aquellas que definen los atributos de calidad de un sistema, como rendimiento, escalabilidad, disponibilidad y seguridad. Estas decisiones influyen directamente en la capacidad del sistema para satisfacer los requisitos tanto funcionales como no funcionales. (Bass et al., 2021).
También son cruciales para definir la estructura y comportamiento de un sistema. Estas decisiones se centran en cómo organizar el código, las responsabilidades de cada módulo y la interacción entre componentes. (Martin, 2017) enfatiza la importancia de mantener la arquitectura flexible y adaptable para permitir cambios futuros sin grandes refactorizaciones.
Las decisiones arquitectónicas se basan en los requerimientos específicos del proyecto y en un análisis detallado del estado actual del sistema. Este proceso ha implicado una estrecha colaboración con las partes interesadas del negocio para asegurar que las decisiones arquitectónicas estén alineadas con las metas y objetivos del mismo. De esta manera, se busca no solo cumplir con los requisitos técnicos, sino también brindar soluciones que impulsen el éxito y la eficiencia del proyecto en su conjunto.

Decisiones de arquitectura	
AD001 - Estilo Arquitectónico	Se definió una arquitectura orientada a comandos y microservicios con el fin de crear un sistema altamente escalable, adaptable y tolerante a fallos. ▪ **Arquitectura basada en Microservicios:** Esta arquitectura permite desarrollar, desplegar y escalar servicios de manera independiente, lo que acelera el ciclo de desarrollo y facilita la adaptación a los cambios en las necesidades del negocio. Cada microservicio está diseñado siguiendo el principio de responsabilidad única, lo que significa que cada servicio tendrá una responsabilidad claramente definida y se encargará de una única función dentro del sistema. Esto no solo mejora la modularidad y facilita el mantenimiento, sino que también permite una mejor gestión de los recursos y la resiliencia del sistema, ya que la falla de un microservicio no afecta a los demás. ▪ **Arquitectura orientada a comandos:** Esta arquitectura permite una mayor separación de responsabilidades y claridad en la ejecución de tareas dentro del sistema. Cada comando representa una acción específica que el sistema debe realizar, lo que facilita la comprensión y el mantenimiento del código. Las ventajas de esta arquitectura incluyen una mayor modularidad, ya que cada comando es independiente y puede ser desarrollado, probado y desplegado de forma aislada. Mejora la trazabilidad y el control sobre las operaciones del sistema, ya que cada comando genera un registro de su ejecución, lo que facilita la auditoría y el análisis de problemas. Esta arquitectura también permite una mejor gestión de la concurrencia y el paralelismo, ya que los comandos pueden ser ejecutados en paralelo sin interferir entre sí, optimizando así el rendimiento y la escalabilidad del sistema.

Tabla 3.30: Decisiones de arquitectura.

Decisiones de arquitectura	
AD002 - Stack	▪ **Framework:** Net Framework ▪ **Unit Test:** XUnit ▪ **Librería scrapping:** Selenium
AD003 - Uso de contenedores	El sistema no usa contenedores ya la orquestación de contenedores agrega complejidad que el negocio no puede asumir actualmente.
AD004 - Monitoreo y gestión	Se utiliza elasticsearch para el almacenamiento de logs y kibana para visualización de los mismos, en esta plataforma se crearon los tableros para monitoreo del sistema. Estos logs solo proporcionarán información del estado del sistema y no se enfocará en indicadores de negocio.
AD005 - Broker de mensajería	Se usa rabbitmq ya que es una cola de mensajería robusta, escalable y que puede implementarse en la nube y on premise.
AD006 - Base de datos	Mysql 8.0.26
AD007 - Repositorio de código	Github. Se utilizará un repositorio por cada componente.
AD008 - API Gateway	Kong API Gateway.
AD009 - Servidor Web y Load Balancer	Se usó IIS como servidor web y se configurará como balanceador de carga.

Tabla 3.31: Decisiones de arquitectura. (Continuación)

Decisiones de arquitectura - Componentes		
ID	**Componente**	**Descripción**
CS001	procesos-api	Centraliza la información de los procesos judiciales del lado de Redjuducial, permite consultar los procesos judiciales pendientes por revisar, registrar actuaciones y actualizar el estado (fecha de revisión) de un proceso judicial.
CS002	extractor-api	Permite la centralización y parametrización de los componentes del sistema de extracción. Permite registrar nodos.
CS003	auth-api	Permite que los usuarios de Redjudicial se autentiquen.
CS004	extractor-publisher	Centraliza la ejecución de comandos (sincronización). Este componente se encarga de consultar los procesos judiciales y publicarlos en la cola para que sean ejecutados por el extractor.
CS005	extractor-subscriber	Se suscribe a la cola para recibir los procesos judiciales (comandos) y a través de scraping descarga la información de RamaJudicial, posteriormente envía la información a extractor-api a través de http
CS006	extractor-checker	Se instala en conjunto con el extractor-subscriber y se encarga de monitorear la salud del subscriber, envía la información a elastic. En caso de fallos tiene la capacidad de reiniciar el subscriber y actualizar la versión de chromedriver.
CS007	notification-service	Servicio en cargado de enviar las notificaciones a los clientes en las horas establecidas.
CS008	extractor-web	Aplicaicón web que permite gestionar gráficamente parámetros y métricas del extractor, permite gestionar procesos judiciales y ver detalles de los mismos.

Tabla 3.32: Decisiones de arquitectura: Componentes.

3.5.2. Tácticas de Arquitectura

Las tácticas de arquitectura permiten abordar desafíos específicos y optimizar aspectos clave del sistema, incluyendo la disponibilidad, el rendimiento, la seguridad y la mantenibilidad. A continuación se explica cómo cada táctica contribuye a la estructura general del sistema y cómo se alinean con los objetivos y requisitos del proyecto.

Táctica	Escenario	Descripción
AT001 - Monitor	RJQS004, RJQS005, RJQS006	A través de un ,componente se monitorea el estado del extractor con el fin de detectar situaciones que puedan generar errores en el sistema, como la versión de chromedriver o la degradación de Ramajuducial
AT002 - Degradation	RJQS004	El sistema reduce ,la cantidad de peticiones simultáneas que realiza cuando Ramajudicial se degrada, de tal manera que pueda seguir funcionando.
AT003 - Retry	RJQS003	El sistema es capaz de reintentar obtener la información de un componente web hasta que este esté listo o se cumpla un límite de tiempo.
AT004 - Redundancia activa	RJQS008	El sistema cuenta con múltiples instancias del extractor-subscriber en diferentes computadores, de modo que si uno falla, otro pueda tomar el relevo de manera inmediata sin interrumpir el servicio
AT005 - Balanceo de carga	RJQS002, RJQS008	El sistema cuenta con múltiples instancias del extractor-subscriber entre las cuales se distribuye la carga según su capacidad.
AT006 - Exception handling	RJQS003, RJQS004, RJQS005, RJQS006	El sistema tiene categorizadas las excepciones que se producen durante la sincronización de tal manera que el checker pueda ejecutar alguna acción dependiendo de la excepción lanzada.
AT007 - Introduce Concurrency	RJQS006	El extractor realiza la sincronización de varios procesos en paralelo, por lo cual una sola instancia puede actualizar varios procesos judiciales al mismo tiempo.

Tabla 3.33: Tácticas de arquitectura Redjudicial

3.5.3. Descripción de la arquitectura

La descripción de la arquitectura según la norma IEEE 42010 es un producto usado para expresar una arquitectura o información tangible proporcionada a las partes interesadas, una metodología estándar para la descripción de la arquitectura de sistemas y software. La elección de utilizar IEEE 42010 se fundamenta en su capacidad para estructurar de manera clara y detallada todos los componentes y relaciones del sistema, asegurando que todos los stakeholders tengan una comprensión común y exhaustiva del proyecto. Esta norma es particularmente útil para abordar y alinear las diversas preocupaciones de los stakeholders mediante la definición de vistas de arquitectura específicas, lo cual es crucial en un proyecto complejo como Redjudicial. iee (2022).
Las vistas de arquitectura son representaciones esenciales que permiten visualizar diferentes aspectos del sistema Redjudicial. Estas vistas facilitan la comprensión, la comunicación y la toma de decisiones a lo largo del ciclo de vida del proyecto. Cada vista proporciona una perspectiva

específica que aborda distintas preocupaciones de los interesados, desde la estructura general del sistema hasta detalles operativos y de despliegue. En la tabla 3.34 se muestra una plantilla con la que se definirán las vistas de arquitectura.

Campo	**Definición**
Interesados	Las personas o roles que tienen interés en la vista específica y sus preocupaciones
Preocupaciones	Los aspectos específicos que interesan a los interesados de esa vista
Punto de Vista	La perspectiva desde la cual se está analizando y documentando la arquitectura
Tipo de Modelo	El tipo de diagrama o modelo utilizado para representar la vista
Modelo	La representación gráfica o el diagrama de la vista
Reglas de Correspondencia	Las reglas para asegurar la consistencia entre diferentes vistas
Justificación	La razón de ser de la vista, explicando por qué es importante y cómo ayuda al proyecto
Identificación de Vista	Un identificador único para la vista, facilitando su referencia

Tabla 3.34: Campos de una Vista de Arquitectura y sus Definiciones

Las vistas de arquitectura ayudan a garantizar que todos los aspectos importantes del sistema sean considerados y documentados, lo que permite una mejor planificación, implementación y mantenimiento del sistema. En el proyecto Redjudicial, estas vistas permiten identificar y mitigar riesgos, asegurar que se cumplen los requisitos funcionales y no funcionales, y asegurar la alineación con los objetivos del negocio.

3.5.3.1. Vista de contexto

La vista de contexto muestra cómo el sistema interactúa con su entorno, incluyendo otros sistemas, personas y procesos externos. Esta vista permite entender los límites del sistema, las interfaces externas y las relaciones con los sistemas y actores externos.

Vista de Contexto	
Interesados	Equipo de Desarrollo, Líderes de Proyecto, Analistas de Negocios, Equipo de Soporte Técnico
Preocupaciones	Entender el alcance y los límites del sistema. Identificar sistemas y actores externos clave. Asegurar que todas las interacciones y dependencias estén claras.
Punto de Vista	Vista de contexto. Usa diagramas de contexto y modelos de interacción a alto nivel.
Tipo de Modelo	Diagrama de contexto.
Modelo	Figura 3.10
Reglas de Correspondencia	Asegura alineación con los objetivos del negocio y requisitos de los usuarios. Cruza referencias con otras vistas para mantener consistencia.
Justificación	Proporciona una comprensión a alto nivel del sistema y su entorno. Ayuda a identificar dependencias e interfaces externas.
Identificación de Vista	RJVW-001

Tabla 3.35: Vista de Contexto Redjudicial

En el diseño se hizo una agrupación por capas con el fin de segmentar las responsabilidades del sistema.

- **Capa de presentación clientes:** Aquellos componentes a través de los cuales interactúan los clientes de Redjudicial.
- **Capa de presentación interna:** Aquellos componentes a través de los cuales interactúan los empleados de Redjudicial.
- **Capa de Datos:** Aquellos componentes a través de los cuales interactúan los usuarios.
- **Capa de microservicios:** Capa de servicios independientes que componen el sistema. Cada microservicio está diseñado para manejar funcionalidades específicas del sistema.
- **Capa de negocio:** componentes involucrados en el proceso de sincronización de procesos judiciales.
- **Capa de Monitoreo:** Componentes que almacena logs del sistema con el fin de verificar su estado.
- **Actores externos:** Servicios externos que son consultados para actualizar los procesos judiciales o enviar notificaciones.

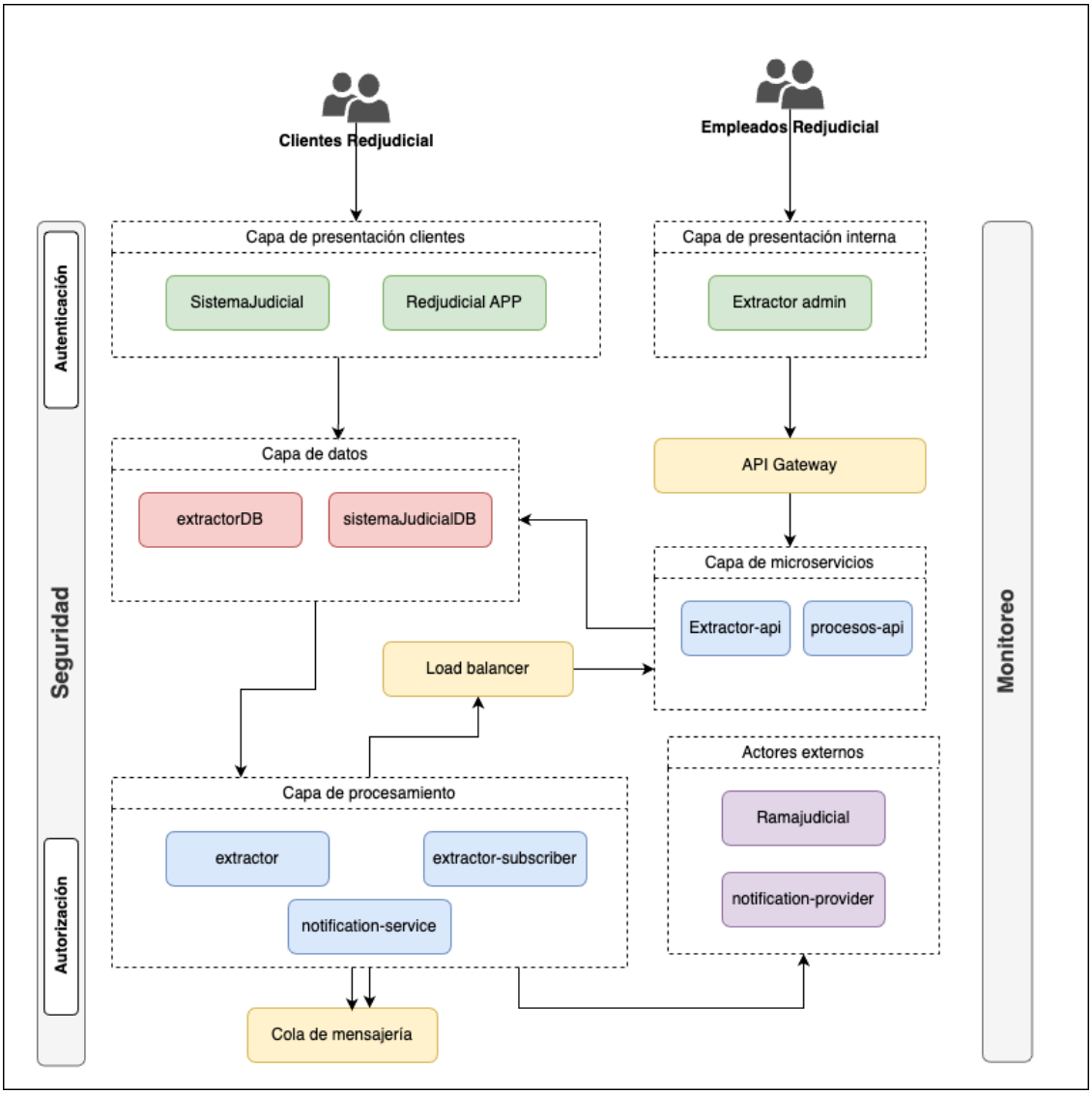

Figura 3.10: Vista de contexto Redjudicial

3.5.3.2. Vista Funcional

La vista funcional se centra en describir las funcionalidades del sistema y cómo estas se organizan y ejecutan dentro de la arquitectura. Esta vista es esencial para entender cómo se cumplen los requerimientos funcionales del sistema a través de sus capacidades, servicios y operaciones ofrecidas. También proporciona detalles sobre los componentes del sistema que implementan estas funcionalidades.

Vista Funcional	
Interesados	Equipo de Desarrollo, Analistas de Negocios.
Preocupaciones	Requisitos funcionales y capacidades del sistema. Cómo interactúan las diferentes funcionalidades dentro del sistema.
Punto de Vista	Vista funcional. Usa diagramas de casos de uso, diagramas de secuencia.
Tipo de Modelo	Diagramas de casos de uso.
Modelo	Figura 3.11
Reglas de Correspondencia	Asegura que se cumplan los requisitos funcionales. Se alinea con el contexto y otras vistas funcionales.
Justificación	Define las principales funcionalidades y sus interacciones. Ayuda a entender el flujo operativo del sistema.
Identificación de Vista	RJVW-002

Tabla 3.36: Vista Funcional Redjudicial

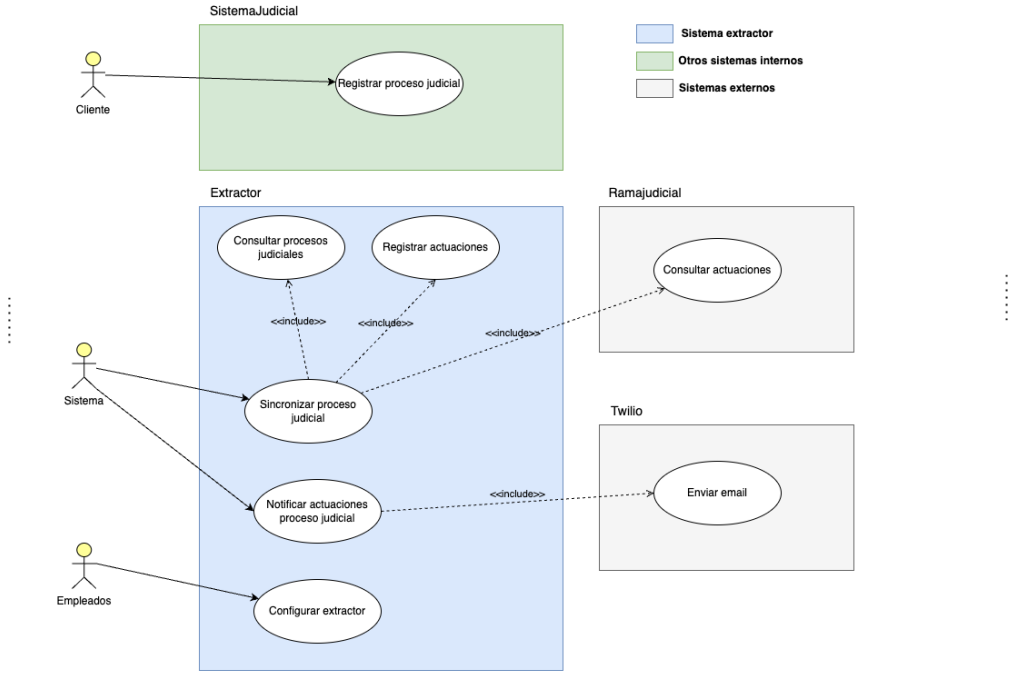

Figura 3.11: Diagrama de casos de uso

3.5.3.3. Vista de datos

La vista de datos del sistema Redjudicial proporciona una representación de cómo se almacenan, gestionan y fluyen los datos dentro del sistema. Esto incluye las entidades de datos principales, las relaciones entre ellas, y los procesos que afectan a los datos.

Vista de Datos	
Interesados	Equipo de Desarrollo, Administradores de Base de Datos, Analistas de Negocios
Preocupaciones	Almacenamiento, gestión y flujo de datos. Asegurar la integridad y accesibilidad de los datos.
Punto de Vista	Modelo ciclo de vida de la información
Tipo de Modelo	Figura 3.12
Modelo	Diagramas
Reglas de Correspondencia	Asegura consistencia de datos en todo el sistema. Se alinea con las vistas funcionales y de despliegue.
Justificación	Ilustra cómo los datos se mueven a través del sistema. Ayuda en el diseño eficiente del almacenamiento y recuperación de datos.
Identificación de Vista	RJVW-003

Tabla 3.37: Vista de Datos para Redjudicial

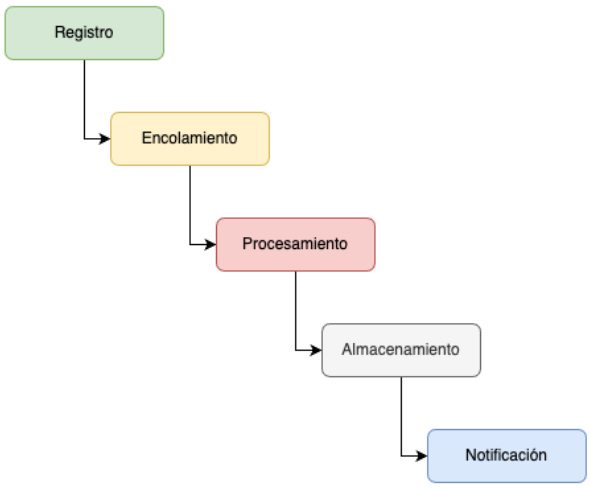

Figura 3.12: Ciclo de vida de la información

Vista de Datos			
Fase	**Descripción**	**Entrada**	**Salida**
Registro	El proceso judicial es registrado en el sistema	Información del proceso judicial	Registro en base de datos
Encolamiento	Los procesos judiciales son encolados para ser sincronizados	Procesos judiciales pendientes por actualizar	Procesos almacenados en la cola de mensajería.
Procesamiento	el sistema consulta Ramajudicial a través de Scraping para obtener y validar las actuaciones recientes.	Proceso judicial encolado.	Actuaciones nuevas.
Almacenamiento	El sistema guarda en la base de datos las actuaciones encontradas.	Actuaciones de Procesos judiciales	Registros en la base de datos
Notificación de actuaciones	El sistema consulta las actuaciones encontradas y registradas en la base de datos por cliente y las envía por email a cada cliente.	Actuaciones nuevas, email de clientes.	Email a cliente.

Tabla 3.38: Ciclo de vida de la información en el extractor Redjudicial

3.5.3.4. Vista de despliegue

La vista de despliegue de Redjudicial se centra en cómo los componentes del sistema serán distribuidos y ejecutados en la infraestructura física y virtual. Inicialmente, el sistema está diseñado para ser implementado en un entorno on premise, aprovechando los servidores físicos existentes en la organización. Esta elección se basa en el aprovechamiento de los recursos de computo con los cuales cuante la empresa y a través de los cuales se quiere distribuir la cargar de procesamiento

Vista de Despliegue	
Interesados	Equipo de Soporte Técnico, Administradores de Sistemas, Equipo de Desarrollo
Preocupaciones	Despliegue físico de los componentes de software. Configuraciones de red e infraestructura.
Punto de Vista	Vista de despliegue. Usa diagramas de despliegue.
Tipo de Modelo	Diagramas de despliegue.
Modelo	Figuras 3.13, 3.14 y 3.15
Reglas de Correspondencia	Asegura que el despliegue físico cumpla con los requisitos del sistema. Cruza referencias con otras vistas para mantener consistencia.
Justificación	Proporciona una visión clara de la arquitectura física del sistema. Ayuda en la planificación y gestión del despliegue e infraestructura.
Identificación de Vista	RJVW-004

Tabla 3.39: Vista de Despliegue para Redjudicial

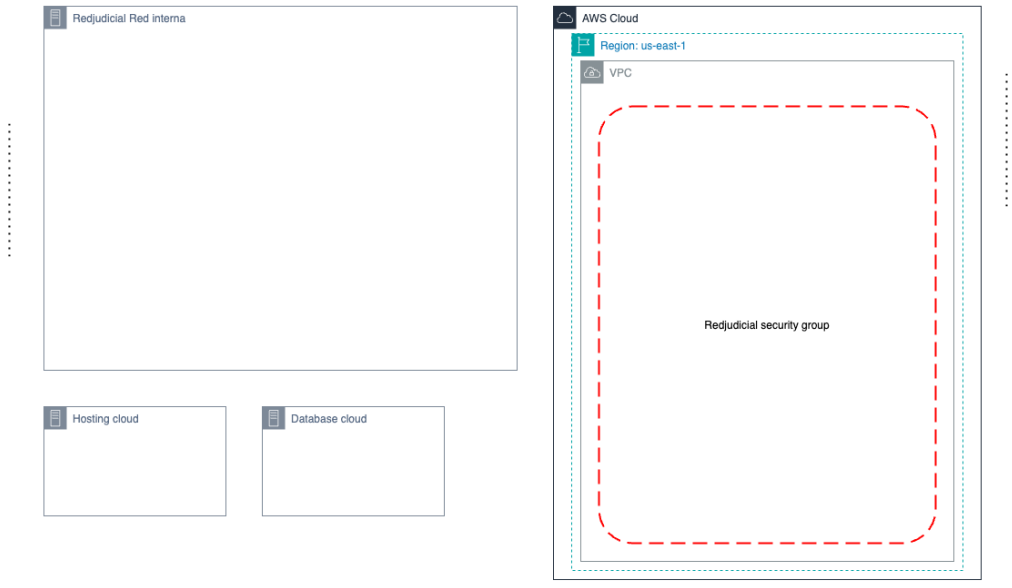

Figura 3.13: Vista de Despliegue en la red

En la figura 3.13 se muestran las redes dentro de las cuales se desplegarán componentes, siendo estas la red interna de Redjudicial y la Red de Amazon web services para implementaciones cloud

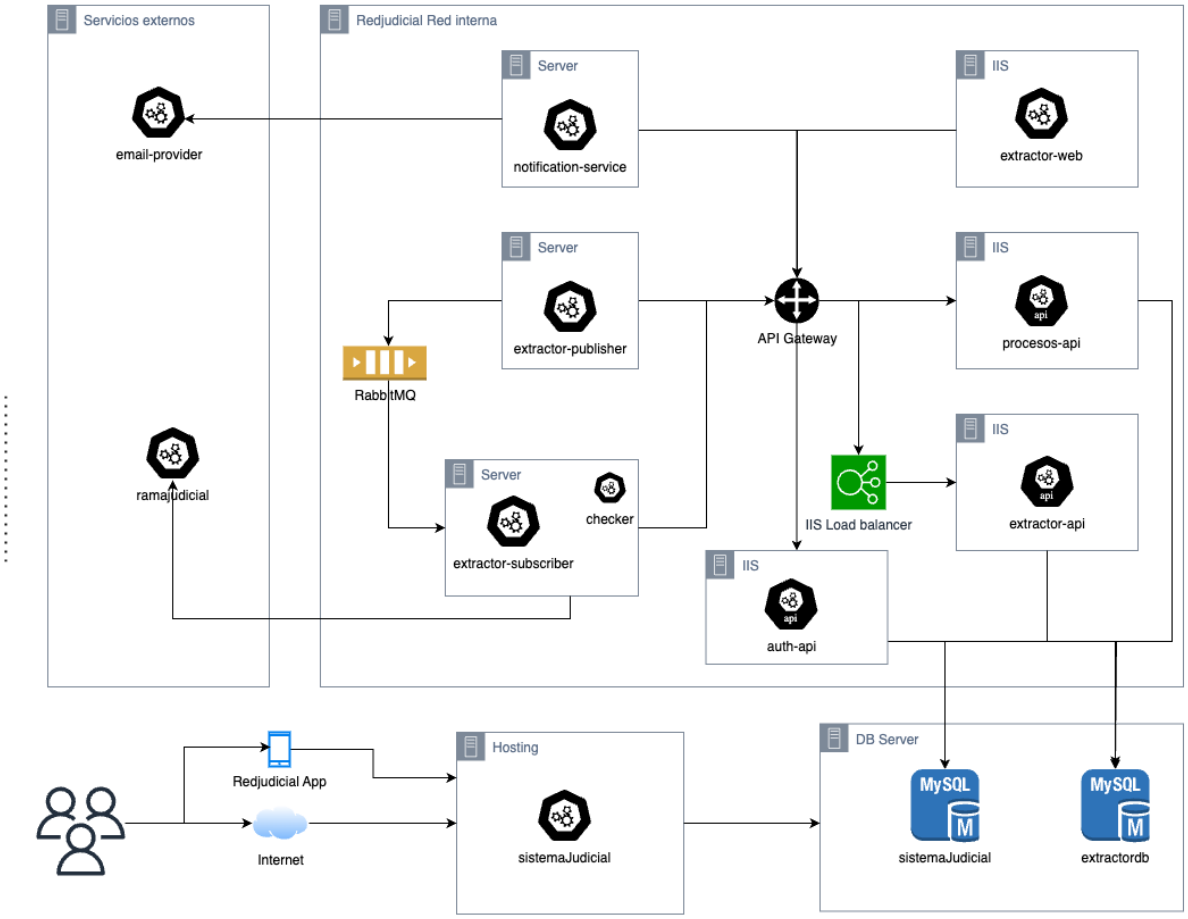

Figura 3.14: Vista de Despliegue en infraestructura local

En la figura 3.14 se muestra como los servidores físicos estarán ubicados en el centro de datos de la organización, asegurando un acceso controlado y monitoreo constante.

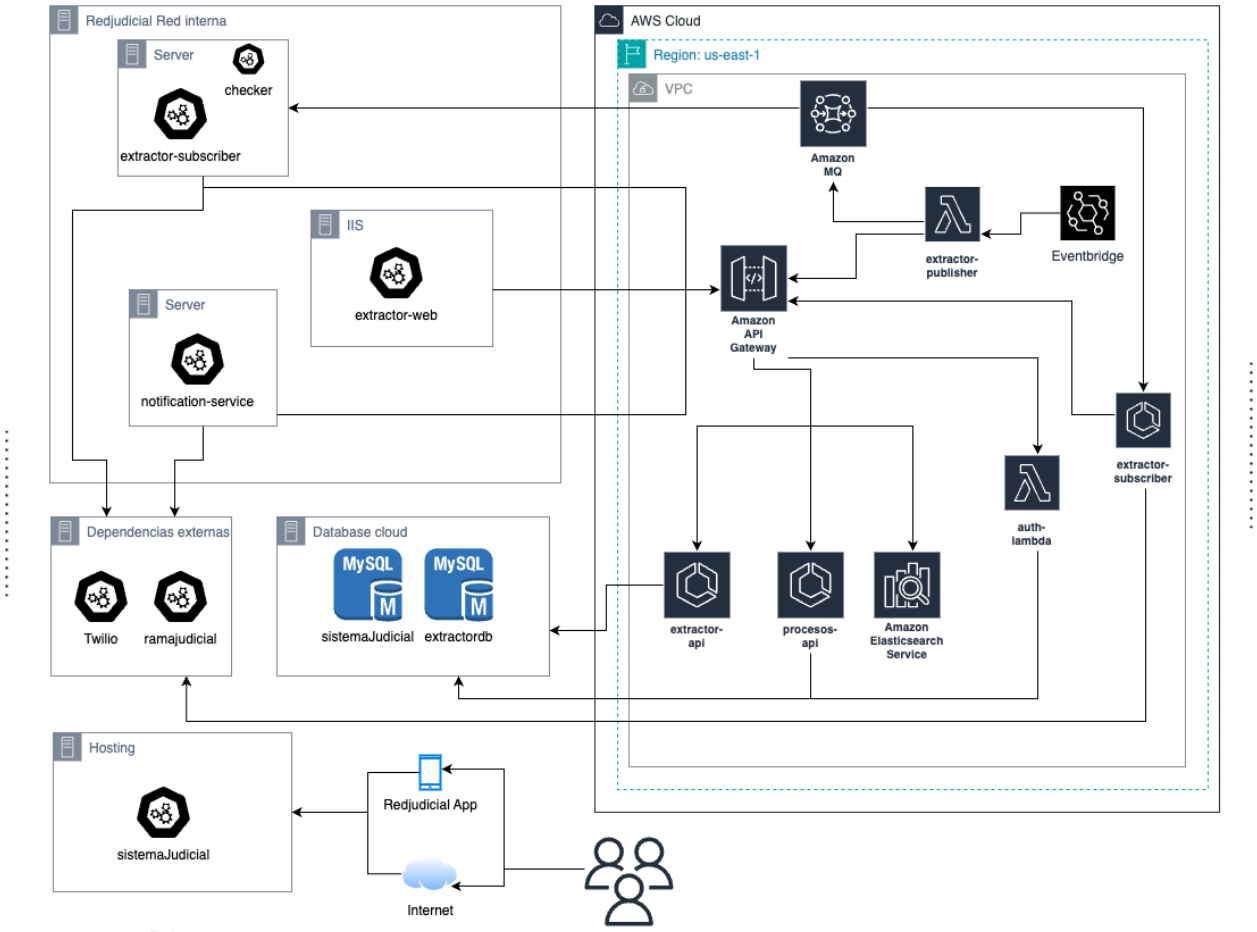

Figura 3.15: Vista de Despliegue en infraestructura híbrida (Local/Cloud)

En la figura 3.15 se muestra cómo sería la implementación dentro de una infraestructura híbrida en la que los componentes que no requieren alta disponibilidad o aquellas de procesamiento scrapping se implementarán dentro de la red interna de la empresa mientras que aquellas que requieren alta disponibilidad se implementarán dentro de la infraestructura de AWS.

3.5.3.5. Vista de concurrencia

La vista de concurrencia se centra en cómo el sistema maneja múltiples procesos ejecutándose simultáneamente. Este enfoque es útil para asegurar que el sistema pueda operar eficientemente bajo cargas de trabajo pesadas y mantener la coherencia de los datos en entornos distribuidos. Permite diseñar estrategias para el control de concurrencia, como el uso de transacciones adecuadas, bloqueos optimizados y gestión de recursos para minimizar conflictos y maximizar el rendimiento del sistema.

Vista de Concurrencia	
Interesados	Equipo de Desarrollo, Arquitectos del Sistema
Preocupaciones	Gestión de concurrencia y sincronización. Asegurar el procesamiento paralelo eficiente.
Punto de Vista	Diagrama de secuencia, Diagrama de concurrencia
Tipo de Modelo	Diagramas de estado. Diagramas de actividad.
Modelo	Figura
Reglas de Correspondencia	Asegura que se aborden los requisitos de concurrencia. Se alinea con las vistas funcionales y de rendimiento.
Justificación	Ilustra cómo el sistema maneja operaciones concurrentes. Ayuda a identificar y resolver posibles problemas de concurrencia.
Identificación de Vista	ID de Vista: RJ-CONCURRENCIA-01

Tabla 3.40: Vista de Concurrencia

En la figura 3.16 se muestra cómo interactúan las diferentes componentes del software a lo largo del tiempo para sincronizar un proceso judicial. Este diagrama es útil para visualizar el flujo de control entre componentes, mostrando cómo se envían y reciben mensajes y cómo se coordina el proceso.

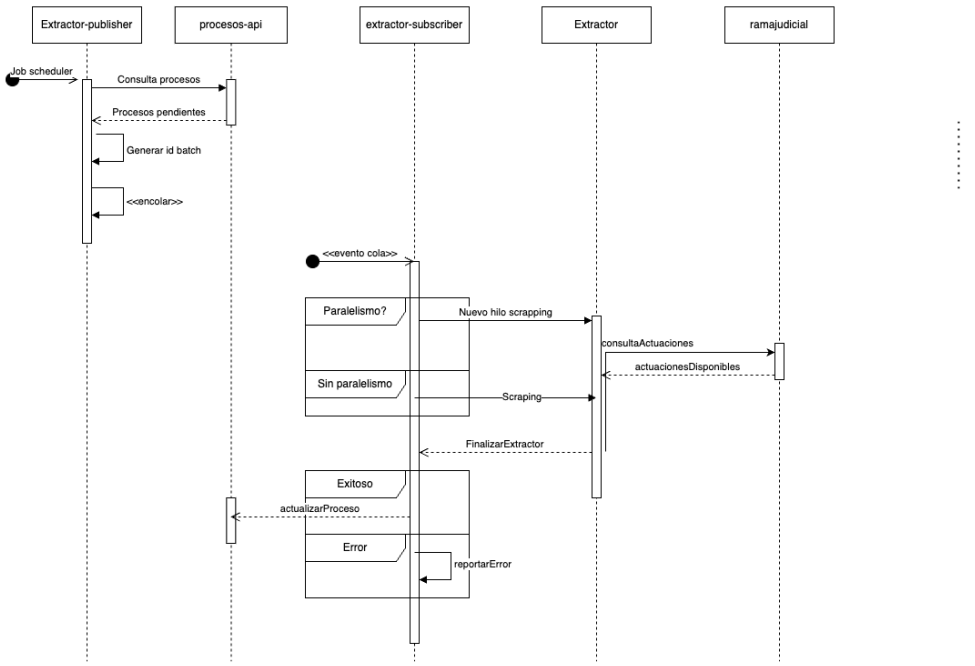

Figura 3.16: Diagrama de secuencia

La figura 3.17 se enfoca en el manejo de hilos o workers dentro del sistema extractor. En este contexto, un worker puede considerarse como un hilo que se encarga de procesar un proceso judicial de manera concurrente, este worker . Esta vista muestra cómo las tareas se distribuyen y procesan simultáneamente, mejorando la eficiencia del sistema.

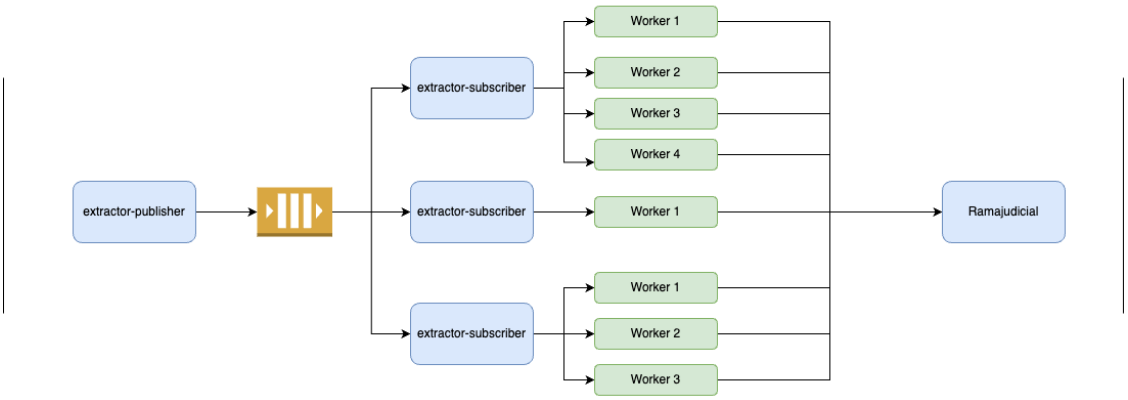

Figura 3.17: Diagrama de concurrencia.

3.6. Operación del sistema

La operación del sistema se refiere a las actividades necesarias para que el sistema funcione correctamente en su entorno de producción. Estas actividades incluyen la configuración, monitoreo, administración, mantenimiento y actualización del software. La operación efectiva del sistema asegura que el sistema cumpla con los requisitos de disponibilidad, rendimiento, fiabilidad y seguridad, lo cual es esencial para el éxito de cualquier proyecto de software.
Actualmente estos procesos no están definidos en Redjudicial, por lo cual es necesario que se definan procesos y prácticas que permitan asegurar el mantenimiento y la operación del sistema.
La decisión de no implementar metodologías DevOps se basa en consideraciones del negocio sobre la complejización tecnológica. Es por esto que se definieron prácticas que permiten garantizar la operación del sistema y sugerencias para una futura implementación de DevOps.

3.6.1. Herramientas de desarrollo

Un aspecto importante en la arquitectura de software de Redjudicial es seleccionar las herramientas de desarrollo adecuadas para asegurar la coherencia y eficacia en el proceso de desarrollo del sistema. Las tecnologías que se definieron han sido escogidas basándose en las restricciones y necesidades específicas de la empresa, considerando que Redjudicial ya emplea predominantemente tecnologías de Microsoft. Esto no solo facilita la integración y el mantenimiento del sistema, sino que también aprovecha las competencias ya desarrolladas por el equipo interno.
Establecer un marco tecnológico coherente, casi como una política empresarial, es vital para asegurar la consistencia en el desarrollo de proyectos futuros. Esto permite mantener un estándar en cuanto a las herramientas y tecnologías utilizadas, lo que se traduce en una curva de aprendizaje más baja para el equipo, mayor estabilidad en la infraestructura y una capacidad más eficiente para resolver problemas que puedan surgir durante el ciclo de vida del software.

3.6.2. Ambientes separados

Redjudicial opera con ambientes (desarrollo, producción y pruebas) separados para asegurar que las diferentes fases del desarrollo y despliegue del software se realicen de manera segura y controlada. Estos ambientes incluyen desarrollo, pruebas y producción. Mantener estos ambientes separados permite que los desarrolladores y testers puedan trabajar sin interferir con la operación en producción, asegurando que los cambios y nuevas funcionalidades sean exhaustivamente evaluados antes de ser implementados en el ambiente de producción.

3.6.3. Despliegue y rollback

El despliegue de nuevas versiones del sistema Redjudicial se realiza siguiendo un proceso estructurado para minimizar el riesgo de interrupciones. Este proceso incluye el uso de scripts y la ejecución del componente a través de consola antes de reemplazar a la versión anterior.

Aunque en la fase actual el despliegue debe hacerse manualmente, se ha implementado un health-check en cada componente con el cual se pueda verificar el estado de este y el extractor-checker se encarga de esto. Al implementar un componente, el usuario debe verificar que éste funcione en el dashboard de Kibana.

3.6.4. Continuidad del negocio

La continuidad del negocio es un aspecto clave en la operación de Redjudicial. El sistema ha sido diseñado para asegurar que los servicios críticos estén siempre disponibles, incluso en situaciones de emergencia o fallo del sistema. Esto incluye la implementación de estrategias de respaldo y recuperación, redundancia en la infraestructura, y procedimientos bien definidos para la restauración rápida de servicios en caso de interrupciones. Asegurar la continuidad del negocio es vital para mantener la confianza de los clientes y garantizar que el servicio de vigilancia de procesos judiciales funcione sin interrupciones, independientemente de las circunstancias externas o internas que puedan afectar el sistema.

3.6.4.1. Respaldo de datos

El sistema funciona con bases de datos en la nube y con replicación, lo que permite tener respaldo en caso de fallas de una instancia; además, se hacen copias de seguridad diarias. Esto se aplica para todo el sistema y no solo al extractor.

3.6.5. Disponibilidad

El extractor no requiere funcionamiento 24/7, por lo cual puede dejar de operar fuera de horario laboral. En caso de que el sistema no pueda seguir operando, se pueden activar instancias en la nube. Cuando se haga la migración a AWS, se dejará una instancia corriendo siempre para que ésta pueda respaldar la operación si las instancias de la red interna dejan de funcionar.

3.6.6. Monitoreo continuo:

El monitoreo continuo es un componente esencial en la gestión de sistemas modernos, como Redjudicial, que requieren alta disponibilidad y rendimiento. Para asegurar el funcionamiento óptimo del sistema y la rápida detección de problemas, se ha implementado una estrategia de monitoreo basada en Elasticsearch y Kibana. Estas herramientas permiten la recopilación, almacenamiento y análisis en tiempo real de los logs generados por el sistema, ofreciendo una visibilidad completa sobre el comportamiento y estado del mismo.
Elasticsearch proporciona un motor de búsqueda y análisis potente, capaz de indexar grandes volúmenes de datos, lo que facilita la rápida recuperación de información relevante. Kibana, por su parte, permite la visualización de estos datos a través de dashboards interactivos y personalizados, facilitando el seguimiento continuo de métricas clave como el rendimiento, el uso de recursos y la detección de anomalías. Este enfoque integral de monitoreo asegura que el equipo técnico pueda

identificar y resolver rápidamente cualquier incidente, garantizando la estabilidad y confiabilidad del sistema Redjudicial.

3.6.7. Herramientas DevOps

Se sugiere el uso de herramientas DevOps para mejorar la eficiencia, automatización y gestión del ciclo de vida del software, aunque no se implementarán en la fase actual del proyecto. Estas herramientas están diseñadas para facilitar la colaboración entre los equipos de desarrollo y operaciones, permitiendo la entrega continua y la integración de software de manera más ágil y confiable. A continuación, se mencionan algunas de las herramientas DevOps recomendadas, sus aplicaciones específicas, y los beneficios que aportarían a la empresa.

3.6.7.1. Jenkins (Integración Continua y Entrega Continua - CI/CD)

Jenkins es una herramienta de automatización que permitiría a Redjudicial implementar pipelines de integración continua. Esto significa que cada vez que se realicen cambios en el código, el sistema podrá compilar, probar y desplegar automáticamente, lo que reduce el tiempo y los errores asociados con los procesos manuales. Esto permitirá agilidad en la entrega de nuevas funcionalidades y mejoras continuas en el sistema, con menor riesgo de introducir errores en producción.

3.6.7.2. Docker (Contenedores)

Aunque en esta fase del proyecto no se implementarán contenedores, Docker podría facilitar el aislamiento de aplicaciones en entornos controlados, lo que permitiría que las aplicaciones se ejecuten de manera consistente en cualquier entorno (desarrollo, pruebas, producción). Con esto se logrará portabilidad de las aplicaciones y facilidad para replicar ambientes de trabajo, mejorando la eficiencia en pruebas y despliegues.

3.6.7.3. Kubernetes (Orquestación de Contenedores)

Si en el futuro se adopta Docker, Kubernetes podría gestionarlo a gran escala, permitiendo el despliegue, escalado y monitoreo de múltiples instancias del sistema. Esto permitirá tener una gestión eficiente de los recursos de infraestructura y escalabilidad automática del sistema según la demanda de los usuarios, lo que sería útil para los picos de carga.

Capítulo 4

Evaluación

4.1. Evaluación del sistema

En este capítulo se realiza la evaluación del proyecto mediante un enfoque integral que incluye una evaluación técnica y de negocio. Para asegurar que el sistema cumpla con los requisitos de calidad establecidos, se utilizará el método de Análisis de Arquitecturas de Software (SAAM), el cual proporcionará una estructura sistemática para evaluar y validar diferentes aspectos de la arquitectura del sistema.

4.1.1. Proceso de evaluación

Para realizar la evaluación se implementaron los componentes básicos del sistema, de tal manera que a través de estos se pudiera medir el desempeño de la arquitectura. Estas pruebas se realizaron en un ambiente similar al de las pruebas del estado inicial del sistema con el fin de comparar los resultados de la nueva implementación con los mismos recursos de la inicial.

- **RabbitMQ:**
- **extractor-subscriber**
- **extractor-publisher**
- **notification-service**
- **extractor-api**
- **procesos-api**
- **Elasticsearch**

Con base en las métricas y logs generados por los componentes implementados y almacenados en Elasticsearch, se realizó la evaluación del desempeño del sistema. En este proceso, se midió la capacidad del sistema para ejecutar las funciones requeridas, se analizaron los tiempos estimados y se compararon con los resultados obtenidos en la versión inicial del sistema.

4.1.2. Evaluación Técnica

La evaluación técnica se centra en revisar la implementación del sistema en términos de su arquitectura, tecnologías utilizadas y la eficiencia del código. Los criterios de evaluación incluyen:

4.1.2.1. Evaluación Funcional

A través de la evaluación funcional se pretende verificar que la arquitectura implementada cumple con los requisitos funcionales del sistema,

RQFN001 - Actualizar proceso judicial	
Descripción:	El extractor debe consultar la información de cada proceso judicial en rama judicial. Si el proceso tiene actuaciones recientes, estas deben registrarse en rama judicial para que puedan ser accedidas por los clientes. Ver tabla 3.6
Estado	Aceptado
Criterios de aceptación	
AC001	El extractor es capaz de consultar los procesos judiciales pendientes por sincronización
CA002	El extractor-subscriber utiliza selenium para consultar la información de Ramajudicial usando scraping
AC003	El extractor-subscriber obtiene las actuaciones del proceso judicial utilizando selenium y es capaz de registrarlas en la base de datos utilizando el extractor-api

Tabla 4.1: Evaluación requerimiento funcional RQFN001

RQFN002 - Notificación clientes	
Descripción:	Una vez al día, en una hora definida por el negocio, el sistema debe notificar a los clientes a través de email las actuaciones que se hayan encontrado para sus procesos. Ver tabla **??**
Estado	Aceptado
Evaluación de Criterios de aceptación	
AC004	El servicio notification-service es capaz de consultar los procesos judiciales pendientes por sincronización desde el extractor-publisher llamando al extractor-api, esto le permite consultar esta información a través de solicitrudes REST
AC005	El servicio utiliza el api del proveedor de email para enviar a los clientes las actuaciones encontradas.

Tabla 4.2: Evaluación requerimiento funcional RQFN002

4.1.3. Evaluación No Funcional

La evaluación no funcional se enfoca en aspectos como el rendimiento, la seguridad, la fiabilidad y la escalabilidad del sistema. Los criterios de evaluación incluyen:

4.1.3.1. Metodología de evaluación

El método SAAM se utilizó para estructurar la evaluación y asegurar que todos los aspectos importantes de la arquitectura sean considerados.

Para la evaluación de los escenarios de calidad se clasificaron en tres tipos (Directos, indirectos y parciales), los cuales determinan si la arquitectura cumple con los escenarios de calidad. Para la evaluación no funcional de los escenarios de calidad se usó el formato de la tabla **??**

- **Escenario Directo:** Un escenario de calidad que se satisface con la arquitectura de software sin modificar el diseño
- **Escenario Indirecto:** Un escenario de calidad que no se satisface con la arquitectura de software y requiere modificar el diseño
- **Escenario:** Aquellos que no dependen totalmente de la arquitectura.

Evaluación Escenarios de Calidad	
Escenario	Identificador del escenario de calidad y título
Tipo	Clasificación (Directo, indirecto, parcial)
Descripción:	Explicación de por qué tiene la clasificación asignada.

Tabla 4.3: Evaluación escenario de calidad RJQS001

4.1.3.2. Evaluación de Escenarios de Calidad

Evaluación Escenario de Calidad RJQS001	
Escenario	RJQS001 Adecuación Funcional Sincronización de procesos judiciales
Tipo	Escenario Directo
Descripción:	El diseño actual garantiza que los procesos judiciales se sincronicen a través de un proceso asíncrono: ▪ extractor-publihser: Consulta los procesos judiciales que deben ser sincronizados. ▪ RabbitMQ: Permite almacenar y encolar los procesos judiciales que se van a actualizar. ▪ extractor-subscriber: Se encarga de consultar las nuevas actuaciones en Ramajudicial y las envía a procesos-api para regisrtrarlas. ▪ procesos-api: Permite consultar los procesos que deben ser sincronizados en la siguiente iteración y registrar las nuevas actuaciones

Tabla 4.4: Evaluación escenario de calidad

Evaluación Escenario de Calidad RJQS002	
Escenario	RJQS002 Rendimiento Sincronización de procesos judiciales
Tipo	Escenario Directo
Descripción:	Este escenario es cubierto ya que un solo nodo del sistema es capaz de procesar varios procesos al mismo tiempo: ▪ extractor-subscriber: Usa paralelismo para procesar en multiples hilos, lo cual permite aprovechar los recursos hardware para sincronizar varios procesos al tiempo. ▪ RabbitMQ: Permite encolar los procesos judiciales para que el subscriber los tome en cuanto se liberan hilos.

Tabla 4.5: Evaluación escenario de calidad

Evaluación Escenario de Calidad RJQS003	
Escenario	RJQS003 Tolerancia a Fallos Sincronización de procesos judiciales
Tipo	Escenario Directo
Descripción:	El extractor-publisher implementa polling pattern lo cual permite que reintente varias veces hasta que un componente web responda o se cumpla un tiempo límite.

Tabla 4.6: Evaluación escenario de calidad

Evaluación Escenario de Calidad RJQS004	
Escenario	RJQS004 Tolerancia a Fallos Sincronización de procesos judiciales
Tipo	Escenario Directo
Descripción:	El diseño actual permite identificar el degrado: ▪ extractor-publihser: Consulta desde el extractor-api el número de procesos que puede procesar el sistema. ▪ extractor-api: Consulta elasticsearch para extraer métricas de disponibilidad del sistema con las cuales se calculará cuántos procesos puede procesar aproximadamente. ▪ extractor-subscriber: Genera excepciones personalizadas para identificar los errores de Ramjudicial, cuenta con un health check para verificar el estado del sistema. ▪ procesos-api: Verifica el estado del extractor-subscriber as través del health check y determina si debe reportar errores de disponibiidad a las métricas de capaciad del sistema ▪ elasticsearch: Almacena logs y métricas de la capacidad del sistema en un periodo de tiempo dado.

Tabla 4.7: Evaluación escenario de calidad

Evaluación Escenario de Calidad RJQS005	
Escenario	RJQS005 Tolerancia a Fallos Sincronización de procesos judiciales
Tipo	Escenario Directo
Descripción:	El sistema permite almacenar logs, entre ellos los errores que se presentan y mostrarlos en kibana lo cual facilita su seguimiento y permite al equipo manejarlos.

Tabla 4.8: Evaluación escenario de calidad

Evaluación Escenario de Calidad RJQS006	
Escenario	RJQS006 Rendimiento Sincronización de procesos judiciales
Tipo	Escenario Directo
Descripción:	Cuando el extractor deja de funcionar por una falla de Ramajudicial o cualquier otra razón podrá recuperarse gracias a que los procesos judiciales se encolan, además el servicio procesos-api consulta los procesos que quedaron pendientes hasta la última iteración realizada.

Tabla 4.9: Evaluación escenario de calidad

Evaluación Escenario de Calidad RJQS007	
Escenario	RJQS007 Interoperabilidad Sincronización de procesos judiciales
Tipo	Escenario Directo
Descripción:	En el diseño se desarrolló una integracióna a Ramajudicial a través de scraping en el componente extractor-subscriber

Tabla 4.10: Evaluación escenario de calidad

Evaluación Escenario de Calidad RJQS008	
Escenario	RJQS008 Escalabilidad Sincronización de procesos judiciales
Tipo	Escenario Directo
Descripción:	El extractor-publisher es el encargado de realizar la sincronización, este puede escalar horizontalmente. El servicio procesos-api también es escalable.

Tabla 4.11: Evaluación escenario de calidad

Evaluación Escenario de Calidad RJQS009	
Escenario	RJQS009 Adecuación Funcional Notificación de actuaciones de procesos judiciales
Tipo	Escenario Directo
Descripción:	El sistema cuenta con un servicio de notificaciones (notification-serivce) que ejecuta jobs que consultan las nuevas actuaciones a través del proceso procesos-api y envía las notificaciones a través del proveedor de email..

Tabla 4.12: Evaluación escenario de calidad

Evaluación Escenario de Calidad RJQS010	
Escenario	RJQS010 Interoperabilidad Notificación de actuaciones de procesos judiciales
Tipo	Escenario Directo
Descripción:	El sistema ya está integrado con el proveedor de envío de emails.

Tabla 4.13: Evaluación escenario de calidad

Evaluación Escenario de Calidad RJQS011	
Escenario	RJQS011 Tolerancia a Fallos Notificación de actuaciones de procesos judiciales
Tipo	Escenario Directo
Descripción:	El sistema maneja los errores que se presentan en el notification-service y las envía a elasticsearch para ser visualizadas.

Tabla 4.14: Evaluación escenario de calidad

4.1.3.3. Métricas - To Be

Para evaluar el rendimiento y la efectividad del sistema después de la implementación de la arquitectura (To-Be), se han utilizado los mismos recursos y condiciones bajo las cuales se realizaron las pruebas del estado anterior (As-Is). Esta metodología asegura que las métricas obtenidas en el estado To-Be sean comparables y permitan una evaluación precisa de las mejoras implementadas. Al mantener constantes las variables de prueba, es posible determinar con mayor claridad el impacto de las optimizaciones y cambios en la arquitectura del sistema, garantizando que los resultados reflejen fielmente el rendimiento y la capacidad del sistema en el entorno proyectado.

- **Idoneidad funcional:** Las métricas de idoneidad funcional permiten si el sistema cumple con los requisitos funcionales pero además establece las medidas con las cuales se determina si

al hacerlo lo hace satisfactoriamente para el negocio. En la tabla ?? se muestran las métricas que permiten concluir que el sistema cumple con el requerimiento y en qué medida

Evaluación métricas de idoneidad funcional			
Métrica	**Valor**	**Evaluación**	**Estado**
Puntualidad de actualización	99.997 %	El sistema es capaz de registrar las actuaciones el mismo día en que son publicadas en Ramajudicial.	Aprobado
Puntualidad de notificación	99.990 %	El sistema notifica las actuaciones que ha encontrado en los horarios establecidos.	Aprobado
Vigilancia	7.75	El sistema es capaz de consultar 7.75 veces en el día cada proceso judicial.	Aprobado

Tabla 4.15: Evaluación métricas de idoneidad funcional

- **Fiabilidad y Tolerancia a Fallos:** Al evaluar la fiabilidad y tolerancia a fallos del sistema se tienen en cuenta métricas que permiten medir si el sistema es capaz de pervenir, manejar o recuperarse ante los fallos.

Evaluación métricas de fiabilidad			
Métrica	**Valor**	**Evaluación**	**Estado**
Tasa de éxito	99.10 %		Aprobado
Tasa de error	0.90 %		Aprobado
Tasa de reintentos	19.39		Aprobado

Tabla 4.16: Evaluación métricas de idoneidad funcional

- **Rendimiento:** Se evalúa cómo el sistema gestiona los recursos o la capacidad que tiene para cumplir con las tareas

Evaluación métricas de rendimiento			
Métrica	**Valor**	**Evaluación**	**Estado**
Tiempo de sincronización	21.06 seg.		Aprobado
Throughput	2412		Aprobado
Uso de memoria RAM	150 MB		Aprobado
Uso de CPU	30 %		Aprobado

Tabla 4.17: Evaluación métricas de idoneidad funcional

4.1.4. Evaluación objetivos de negocio

La evaluación de los objetivos del negocio fue un paso crucial en el análisis del éxito del proyecto Redjudicial. Aunque no se compartirán cifras específicas de la empresa, esta evaluación permite asegurar que el proyecto cumple con las metas estratégicas establecidas por la organización. Al revisar y medir el cumplimiento de estos objetivos, garantizamos que el sistema no solo se alinea con las necesidades operativas y técnicas, sino que también respalda la visión y misión de la empresa. Esta alineación es fundamental para asegurar que los recursos invertidos en el proyecto se traduzcan en beneficios tangibles para Redjudicial, contribuyendo al crecimiento sostenible y la competitividad en su sector.

Objetivo	**Meta**	**Valor alcanzado**	**Cumplimiento**
Mejorar la Eficiencia Operativa	100 %	100 %	100 %
Aumentar la confiabilidad del sistema reduciendo el número de errores	5 %	0.9 %	550 %
Asegurar el crecimiento de la operación	100 %	57.888	100 %
Aumento en la Precisión de las Notificaciones	1 día	1 día	99.997 %

Tabla 4.18: Evaluación Objetivos de Negocio. Fuente: Elaboración propia.

A través de las métricas el sistema pudo establecer la capacidad que el extractor tiene para procesar procesos judiciales y de esta manera establecer cuánto crecería la operación y así mismo los costos. Aunque por motivos de confidencialidad los costos de operación no pueden se publicados.

Capítulo 5

Conclusiones

En esta sección se presentan las conclusiones a las que se llegó durante el desarrollo de este proyecto partiendo desde los objetivos. Es importante mencionar que para la realización de este proceso se hizo una tarea previa de contextualización en la que se abordaron los requerimientos del sistema y los objetivos del negocio, para que así el análisis, definiciones, diagnósticos y propuestas estén enfocados en satisfacer los requerimientos del negocio.

5.1. Análisis del sistema actual

Tras realizar un análisis exhaustivo de los procesos actuales de obtención de datos de la Rama Judicial, se identificaron varios problemas que afectan tanto al rendimiento como a la fiabilidad del sistema de Redjudicial.
Estas falencias están relacionadas con la arquitectura del sistema actual, ya que Redjudicial cuenta con un monolito que, si bien es capaz de sincronizar procesos judiciales, sufre una gran cantidad de errores en el proceso, además de que no tiene la capacidad de procesar el volumen de procesos judiciales de Redjudicial.

- El sistema, al ser un monolito no permite escalar de manera eficiente, para poder hacerlo debe escalar todo el sistema, lo cual es costoso e ineficiente, por lo tanto Redjudicial no puede alcanzar el volumen requerido de procesos judiciales por día.
- El sistema es difícil de mantener y evolucionar ya que cada cambio que se realice requerirá un despliegue de todo el sistema además de que la adopción de nuevas tecnologías afectaría todo el sistema por no ser modular.
- El uso de paralelismo aunque permite aumentar la capacidad del sistema también le agrega complejidad, lo que lo hace más difícil de mantener y monitorear en un monolito. Otro inconveniente con el uso de paralelismo es que el sistema puede superar la capacidad de procesar solicitudes de Ramajudicial, incrementando el número de errores que pueden aumentar el tiempo de indisponibilidad del sistema e incluso dejarlo totalmente fuera de línea ya que los errores son más complejos de manejar.
- El sistema no cuenta con mecanismos para gestionar fallos ni de monitoreo. En este punto se identificaron también aquellos errores más comunes y que afectan el funcionamiento del sistema, el principal de ellos está asociado a fallas por la dependencia a Ramajudicial, lo cual puede inducir a errores por la degradación o caída.

Una de las soluciones temporales que el negocio implementó fue el uso de un script en Python que solo automatiza el proceso de scraping, pero sigue requiriendo transcripción manual y, además, la información que consulta no es tan precisa, por lo cual debe ser filtrada.
A través de este análisis se concluyó que son estas falencias las que no permiten que el proceso se pueda automatizar y escalar para cumplir con las necesidades del negocio; por ende, un cambio en este diseño fue la solución para este problema.

5.2. Diseño de arquitectura

La implementación de una arquitectura modular basada en microservicios y orientada a comandos permitió resolver los problemas encontrados en la versión inicial del sistema de Redjudicial, de tal manera que el sistema cumpliera con el objetivo del negocio (actualizar procesos judiciales).
Uno de los puntos más importantes o de los cambios más significativos en esta arquitectura fue la aplicación del principio de responsabilidad única (Martin, 2017), a través de la separación se simplifica el diseño de tal manera que el sistema sea comprensible, mantenible y pueda evolucionar.
En el diseño actual, tratar cada tarea como un comando permite que el sistema se pueda comunicar eficientemente y ejecutar la actualización de cada proceso judicial como una entidad independiente, facilita agregar mecanismos de reintento o de compensación en caso de fallos. Si un comando falla, puede volver a intentar sin afectar el resto del sistema, lo cual es esencial para que el sistema pueda ejecutar un gran volumen de procesos.
Las tácticas implementadas como reintento, monitoreo y concurrencia permiten que el sistema pueda funcionar con un mejor desempeño e interacción entre los componentes. Todo esto permitió que el sistema, además de satisfacer las necesidades de la operación de Redjudicial, también sea adaptable y pueda evolucionar.

5.3. Gestión de errores y recuperación del sistema

Teniendo en cuenta que la principal causa de fallos del sistema es la degradación o caída de la disponibilidad de Ramajudicial, se implementó un sistema a través del cual se monitorea el desempeño y se reducen o incrementan los llamados según el comportamiento de Ramajudicial, de esta manera aunque el sistema reduce la capacidad de procesos a ejecutar en simultáneo, logra garantizar que Ramajudicial esté disponible y el sistema pueda continuar operando, adicionalmente el sistema es capaz de monitorear su propio estado de tal manera que en algunos casos es capaz de ejecutar comandos predefinidos para reparar el sistema y en caso de no ser posible resolverlo, el sistema es capaz de reportarlo.

5.4. Evaluación y validación de la aquitectura

La evaluación del diseño de arquitectura permitió validar que la arquitectura satisfacía las necesidades del negocio, aplicando la metodología SAAM se pudo analizar desde una perspectiva crítica la calidad y robustez del sistema. Como resultado de esta evaluación se pudo concluir que el sistema al ser escalable y modular pudo cubrir la operación requerida ahora y a futuro por Redjudicial y de esta manera reducir los procesos manuales. Las métricas del sistema mostraron una mejora del 1300 % en la capacidad operativa del sistema.

Capítulo 6

Anexos

A continuación se relacionan los documentos anexos de este proyecto. Estos se generaron usando el enfoque AS-IS TO-BE; en estos se hace una definición del sistema en su estado inicial (AS-IS) y el estado propuesto como solución al problema planteado en este proyecto (TO-BE)

6.1. Diseño As-Is

6.1.1. Redjudicial Diseño As-IS

En este documento se detallan los componentes actuales de Redjudicial, su arquitectura, procesos y las problemáticas identificadas, como limitaciones en rendimiento, escalabilidad y seguridad. Esta parte del documento ofrece una visión clara de cómo opera el sistema actualmente y cuáles son sus deficiencias. Redjudicial Diseño As-IS

6.1.2. Resultados de pruebas As-Is

En este documento se recolectan y tabulan los resultados de las pruebas realizadas sobre el sistema en su estado inicial. Para esto, se evaluaron las transacciones y se guardaron los datos relevantes. El documento contiene las siguientes hojas:

- Resumen de las métricas
- Total de procesos registrados para la prueba, el identificador de los procesos es un identificador interno.
- Transacciones registradas, el proceso judicial y las métricas del mismo.
- Errores identificados durante la prueba
- Datos del servidor en el que se realizaron las pruebas.

Resultados de pruebas As-Is

6.2. Diseño To-Be

6.2.1. Redjudicial Diseño To-Be

Este es el documento final de arquitectura en el que se describe el diseño propuesto para mejorar el sistema. Se detallan los cambios arquitectónicos, mejoras en los procesos de automatización, sincronización y seguridad, y cómo se abordarán las limitaciones actuales. Este diseño propuesto tiene como objetivo alinear el sistema con los objetivos de negocio, mejorar su escalabilidad y garantizar su mantenimiento y evolución a largo plazo. Redjudicial Diseño To-Be

6.2.2. Resultados de pruebas To-Be

En este documento se recolectan y tabulan los resultados de las pruebas realizadas sobre el diseño propuesto. Para esto, se evaluaron las transacciones y se guardaron los datos relevantes. El documento contiene las siguientes hojas:

- Resumen de las métricas
- Total de procesos registrados para la prueba, el identificador de los procesos es un identificador interno.
- Transacciones registradas, el proceso judicial y las métricas del mismo.
- Errores identificados durante la prueba
- Datos del servidor en el que se realizaron las pruebas.

Resultados de pruebas To-Be

Bibliografía

(2022). Ieee/iso/iec international standard for software, systems and enterprise–architecture description. *ISO/IEC/IEEE 42010:2022(E)*, pages 1–74.

(2023). ISO/IEC 25010:2023 - Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models.

Bass, L., Clements, P., and Kazman, R. (2012). *Software Architecture in Practice (SEI Series in Software Engineering)*. Addison-Wesley Professional, 3 edition.

Bass, L., Clements, P., and Kazman, R. (2021). *Software Architecture in Practice*. Addison-Wesley Professional, 4 edition.

Cavallo, A. and Rigobon, R. (2016). The billion prices project: Using online prices for measurement and research. *Journal of Economic Perspectives*, 30(2):151–78.

Cohn, M. (2010). *User Stories Applied: For Agile Software Development*. Addison-Wesley Professional.

Deloitte (2023). La era de la automatización implementación de robotics en las organizaciones. Technical report, Deloitte.

Fowler, M., Rice, D., Foemmel, M., Mee, R., and Stafford, R. (2002). *Patterns of Enterprise Application*.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1996). Design patterns: Elements of reusable software. *Addison-Wesley Professional Computing Series*.

Garlan, D. and Shaw, M. (1993). An introduction to software architecture. *Advances in Software Engineering and Knowledge Engineering*, 1(1):1–39.

Garlan, D., Shaw, M., and Zelkowitz, M. (2004). *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall PTR.

Ionita, M. T., Hammer, D. K., and Obbink, H. (2002). Scenario-based software architecture evaluation methods: An overview. In *Workshop on methods and techniques for software architecture review and assessment at the international conference on software engineering*, pages 1–12. Citeseer.

Jain, M., Vaish, S., Patil, M., and Anant, G. M. (2019). Data extraction and sentimental analysis from "twitter" using web scrapping. *International Journal of Engineering and Advanced Technology*, 9(1):6451 – 6455. Cited by: 0; All Open Access, Bronze Open Access.

Jones, R. (2012). *Metodología As-Is To-Be: Herramientas para la mejora de procesos.* Editorial Empresarial.

Judicial, R. (2023). *Manual de Usuario CPNU.* Accedido el 23 de agosto de 2023.

Kazman, R., Klein, M., and Clements, P. (2000). Atam: Method for architecture evaluation. Technical Report CMU/SEI-2000-TR-004, Software Engineering Institute, Carnegie Mellon University.

Kim, C. G. (2019). An implementation and performance evaluation of fast web crawler with python. *Department of Computer Science, Namseoul University.*

Kim, G., Humble, J., Debois, P., and Willis, J. (2016). *The DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations.* IT Revolution Press.

Martin, R. C. (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design.* Anaya Multimedia.

Richards, M. A. and Ford, N. (2020). *Fundamentals of Software Architecture: An Engineering Approach.* O'Reilly Media, Inc.

Rozanski, N. and Woods, E. (2011). *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives.* Addison-Wesley Professional.

Schwaber, K. and Sutherland, J. (2017). *La Guía Definitiva de Scrum: Las Reglas del Juego.* Scrum.org.

Sutherland, J. and Altman, J. (2014). *Beyond the Daily Standup: Accelerating the Agile Enterprise.* Addison-Wesley Professional.

Tanenbaum, A. S. and van Steen, M. (2023). *Distributed Systems: Principles and Paradigms.* Pearson.

Taylor, R. N. (2021). *The history of software engineering.* CRC Press.

You, J., Lee, J., and Kwon, H.-Y. (2021). A complete and fast scraping method for collecting tweets. In *2021 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 24–27.