Universidad Pontificia Bolivariana

Facultad de Ingenierías - Escuela de Informática

Introducción a la ingeniería de software

Reingeniería de un Sistema Legacy:

Un Estudio de Caso de Transformación Digital

Presentado por:

Miguel Ángel Motta Barrero

Programa:

Ingeniería de Software

Profesor:

Juan Pablo Torres Pabón

Semestre:

2025-2

Modalidad:

Virtual

Información de la propuesta para el estudio de caso

-Objetivo del Estudio

Este estudio de caso hipotético que elegi busca consolidar la comprensión de los procesos de desarrollo de software y fortalecer las habilidades analíticas y de toma de decisiones, a través de los siguientes objetivos:

- **Analizar los desafíos de la reingeniería de sistemas legacy**, identificando problemas técnicos, operativos y organizacionales.
- **Evaluar estrategias y metodologías de modernización de software**, considerando sus ventajas, desventajas y aplicabilidad.
- **Proponer una solución integral y justificada para la transformación del sistema obsoleto**, incluyendo un plan de implementación y métricas de éxito.
- **Desarrollar la capacidad de tomar decisiones informadas** frente a problemas complejos de desarrollo de software, integrando la gestión de riesgos y los impactos en el negocio.

Ya que la modernización de software busca eficiencia, escalabilidad y adaptación a las nuevas demandas del mercado.[1] Los beneficios esperados de una reingeniería exitosa incluyen mayor eficacia, costos reducidos, transparencia mejorada, plazos de comercialización más cortos, innovación incrementada, mayor calidad del producto y una mejora general del rendimiento y la seguridad.[2, 3] por ejemplo con el siguiente contexto.

Contexto (Caso Hipotético)

presentando un escenario hipotético que simula una situación recurrente en el entorno empresarial. La empresa "Logística Ágil S.A.", una compañía de tamaño mediano con 15 años de trayectoria en distribución de productos perecederos, enfrenta una encrucijada tecnológica, su operación central depende de un sistema de gestión de inventarios y rutas desarrollado a principios de los años 2000, construido con tecnologías que hoy se consideran obsoletas (como lo son COBOL y una base de datos jerárquica propietaria), este sistema ha sido el pilar de sus operaciones, pero actualmente presenta serias limitaciones para adaptarse a las dinámicas del mercado moderno y a las ambiciones de expansión de la empresa, es como tener un DeLorean: funcionó de maravilla en su época, pero hoy en día, ¿quién lo conecta a la nube?

La modernización es necesaria no solo por la antigüedad del sistema, sino por una exigencia empresarial. La compañía busca expandirse a nuevos mercados y optimizar sus cadenas de suministro para mantener su ventaja competitiva. Los sistemas heredados suelen obstaculizar la adopción de innovaciones y la eficiencia operativa, resultando en procesos manuales y redundantes.[4] La reingeniería es una respuesta estratégica a presiones competitivas y oportunidades de crecimiento, buscando eficiencia, escalabilidad y adaptación al mercado.[1]

Actores involucrados

La reingeniería de un sistema legacy impacta a múltiples grupos dentro y fuera de la organización. por ejemplo:

- **Usuarios Finales (Departamentos de Operaciones y Logística):** Son los empleados que interactúan diariamente con el sistema para la gestión de inventarios, pedidos y rutas. Son los más afectados por la deficiente experiencia de usuario (UX) y los fallos, lo que impacta su productividad y moral.[4] Su frustración es un motor clave para el cambio, casi como si estuvieran atrapados en un bucle infinito de carga!
- **Departamento de TI:**
  - **Desarrolladores Legacy:** Un pequeño equipo de programadores con conocimiento profundo, pero a menudo no documentado, del código COBOL. Su experiencia es invaluable, pero su limitada exposición a tecnologías modernas puede generar resistencia.[4] Su retención y capacitación son críticas. Son los "guardianes de la galaxia" del código antiguo.
  - **Administradores de Sistemas:** Responsables de mantener la infraestructura de hardware y software obsoleta. Enfrentan desafíos constantes en soporte, seguridad y disponibilidad de piezas.
  - **Personal de Soporte Técnico:** Primer punto de contacto para usuarios, gestionando quejas, incidentes e interrupciones.
- **Gerencia y Dirección (CEO, CFO, Director de Operaciones):** Responsables de las decisiones estratégicas, asignación de presupuestos y supervisión de la eficiencia y rentabilidad. Su interés principal es la eficiencia operativa, escalabilidad y competitividad.[1] Son los principales impulsores de la modernización, buscando que la empresa no se quede en la "Edad de Piedra" digital.
- **Clientes y Proveedores:** Aunque no interactúan directamente con el sistema interno, su experiencia se ve afectada por su eficiencia. Retrasos, errores o falta de visibilidad impactan negativamente su satisfacción.[4] La modernización busca mejorar la satisfacción del cliente y la eficiencia de la cadena de suministro.
- **Expertos Externos/Consultores:** Aliados estratégicos en la evaluación, planificación y ejecución de la reingeniería. Aportan conocimientos especializados en tecnologías modernas y gestión de proyectos. La selección de un proveedor experimentado es crucial.[5]

La reingeniería no es solo tecnológica, sino un desafío socio-técnico que involucra gestión del cambio, capacitación y alineación de intereses. Los actores pueden tener prioridades distintas (ej., gerencia busca costos y agilidad; usuarios, usabilidad y estabilidad). al reconocer estas diferencias haría que una estrategia de comunicación y gestión del cambio más efectiva.

Problemas y desafíos

La obsolescencia del sistema legacy de Logística Ágil S.A. genera múltiples desafíos

interconectados que impactan negativamente en toda la organización.

Los problemas del sistema heredado son extensos y multifacéticos, afectando desde la seguridad hasta la capacidad de innovación:

- **Vulnerabilidad a la Seguridad:** El sistema carece de actualizaciones de seguridad y parches, haciéndolo vulnerable a ciberataques y brechas de datos. Esto pone en riesgo la información crítica, exponiendo a la empresa a riesgos legales y de reputación.[4]
- **Altos Costos de Mantenimiento:** Mantener un sistema obsoleto es muy costoso debido a la escasez de piezas de repuesto y la dificultad para encontrar expertos en tecnologías antiguas como COBOL. Las reparaciones son lentas y caras.[4, 1] A veces, parece que cuesta más mantenerlo vivo que construir uno nuevo desde cero.
- **Incompatibilidad con Nuevas Tecnologías:** El sistema funciona como un silo, incapaz de integrarse con software moderno (e-commerce, análisis de datos, CRM). Esto restringe la adopción de innovaciones y la eficiencia operativa, forzando procesos manuales y redundantes.[4]
- **Baja Escalabilidad y Flexibilidad:** El diseño original del sistema no previó el volumen de operaciones actual ni la adaptación a las demandas cambiantes del negocio (nuevos productos, expansión). Esta rigidez obstaculiza el crecimiento y la respuesta rápida al mercado.[4, 1, 6].
- **Riesgo Elevado de Fallos del Sistema y Tiempos de Inactividad:** La infraestructura envejecida es propensa a fallos inesperados y tiempos de inactividad prolongados. Estas interrupciones paralizan operaciones críticas, resultando en pérdidas financieras y afectando el cumplimiento de compromisos.[4].
- **Dificultades en la Gestión y Análisis de Datos:** La gestión de datos es ineficiente y propensa a errores por la estructura anticuada del sistema. La falta de integración con herramientas modernas dificulta el acceso y análisis de datos, impidiendo decisiones estratégicas basadas en información precisa.[4]
- **Experiencia de Usuario Deficiente (UX):** Las interfaces y funcionalidades anticuadas del sistema resultan en una experiencia de usuario deficiente para los empleados, afectando su productividad y moral.[4] Los usuarios se sienten como si estuvieran usando una máquina de escribir en la era de los smartphones.
- **Cumplimiento Normativo:** El sistema tiene dificultades para cumplir con normativas y estándares actuales (privacidad de datos, trazabilidad). Esto expone a la empresa a riesgos legales y regulatorios, incluyendo multas.[4] Un verdadero dolor de cabeza legal.
- **Deuda Técnica Elevada:** La acumulación de parches y soluciones rápidas sin refactorización o documentación ha creado un código complejo, indocumentado y frágil. Esta "deuda" hace que cualquier modificación sea difícil, lenta y propensa a errores, aumentando el riesgo del proyecto.[1] Es la "hipoteca" del software, y está creciendo.

Estos problemas no son aislados; forman una red compleja donde cada deficiencia exacerba a las demás (ej., deuda técnica dificulta seguridad, aumentando vulnerabilidad). Una solución efectiva debe ser holística, abordando las causas subyacentes. Además de los costos directos, existen costos "ocultos" como la pérdida de oportunidades o la baja moral de empleados,

perjudiciales para la viabilidad a largo plazo, para una mejor visualización de los problemas y sus impactos, hice la siguiente tabla:

Tabla 1: Problemas Comunes en Sistemas Legacy y su Impacto

Problema Identificado	Descripción Breve	Impacto en el Negocio
Vulnerabilidad a la Seguridad	Falta de actualizaciones y parches de seguridad.	Riesgo de brechas de datos, multas regulatorias, pérdida de confianza del cliente.
Altos Costos de Mantenimiento	Dificultad para encontrar expertos y piezas de repuesto para tecnologías obsoletas.	Gastos operativos elevados, asignación ineficiente de recursos.
Incompatibilidad con Nuevas Tecnologías	Incapacidad para integrarse con software moderno.	Procesos manuales y redundantes, limitación para la innovación y eficiencia.
Baja Escalabilidad y Flexibilidad	Sistema no diseñado para el volumen o las necesidades actuales del negocio.	Obstaculiza el crecimiento, impide la rápida adaptación al mercado.
Riesgo Elevado de Fallos y Tiempos de Inactividad	Infraestructura envejecida propensa a interrupciones.	Pérdidas financieras por interrupción de operaciones, impacto en la reputación.
Dificultades en la Gestión de Datos	Ineficiencia y errores en el manejo y análisis de datos.	Obstaculiza la toma de decisiones basada en datos, información desactualizada.
Experiencia de Usuario Deficiente (UX)	Interfaces y funcionalidades anticuadas.	Baja productividad de los empleados, insatisfacción laboral, errores operativos.
Cumplimiento Normativo	Dificultad para adherirse a regulaciones actuales.	Riesgos legales, multas, sanciones.

Deuda Técnica Elevada	Acumulación de parches y soluciones rápidas sin refactorización.	Dificultad y riesgo en cualquier modificación, sobrecostos y retrasos en proyectos.

Causas

Los problemas del sistema legacy de Logística Ágil S.A. son resultado de decisiones históricas, falta de inversión y dinámicas organizacionales:

- **Decisiones de Diseño Originales y Arquitectura Monolítica:** El sistema fue concebido como un monolito, con componentes fuertemente acoplados y dependencias complejas. Esta arquitectura dificulta la modularidad y la modificación de funcionalidades, haciendo la adopción de nuevas tecnologías un desafío.[7, 8] Es como un castillo de naipes: si mueves una, todo se viene abajo.
- **Falta de Inversión Continua en Mantenimiento Evolutivo:** No se invirtió en actualizaciones periódicas, refactorización o modernización incremental. La inversión se centró en añadir funcionalidades rápidas y reactivas, en lugar de mejorar la base tecnológica. Esta es una causa fundamental de la obsolescencia y los problemas.
- **Dependencia de Tecnologías Propietarias y Obsoletas:** El uso de lenguajes y bases de datos menos comunes o sin soporte oficial ha creado un "bloqueo tecnológico". Esto dificulta la integración con sistemas modernos y genera escasez de talento, aumentando los costos de soporte y mantenimiento.[4] Es como tener un coche que solo usa gasolina de una marca que ya no existe.
- **Acumulación de Deuda Técnica:** La preferencia por soluciones rápidas y parches sobre un diseño robusto, junto con la falta de documentación, pruebas automatizadas y estándares, ha resultado en un código frágil y difícil de modificar. Esta deuda técnica se ha acumulado, haciendo cada cambio de alto riesgo.[1, 7, 9] La factura sigue creciendo, y no hay quien la pague.
- **Falta de Comunicación y Colaboración Interdepartamental:** La comunicación deficiente entre TI, usuarios y gerencia ha desalineado expectativas, resultando en requisitos incompletos o soluciones subóptimas. Esta brecha es un factor recurrente en los desafíos de desarrollo de software.[9] A veces, parece que hablan idiomas diferentes.
- **Gestión de Proyectos Inadecuada:** Estimaciones de tiempo y presupuesto erróneas, falta de control del alcance ("scope creep") y supervisión ineficaz de gastos han contribuido a la situación actual. Una planificación deficiente puede desviar el proyecto del presupuesto y ámbito.[5, 9] Un proyecto sin brújula.
- **Resistencia al Cambio Organizacional:** El temor a la interrupción, la aversión al riesgo, la falta de comprensión de los beneficios de la modernización o la inercia cultural han pospuesto la reingeniería, permitiendo que los problemas se agraven. "Si no está roto, no lo arregles", hasta que explota.

Comprender estas causas subyacentes es fundamental. No es solo la antigüedad del código,

sino un conjunto de factores técnicos y organizacionales que se han reforzado. Reconocer que los problemas de los "sistemas legacy" tienen raíces organizacionales y humanas, no solo técnicas, es crucial. Una reingeniería exitosa requiere experiencia técnica, liderazgo fuerte, gestión del cambio y colaboración interdepartamental.

Impacto

Mantener un sistema legacy obsoleto tiene repercusiones considerables, con impactos tangibles e intangibles en Logística Ágil S.A., lo que subraya la urgencia de una intervención.

- **Pérdidas Financieras Directas:** Los altos costos de mantenimiento y operación del sistema legacy representan una constante pérdida de recursos financieros.[4] Las interrupciones y fallos resultan en pérdidas de ingresos directas por la imposibilidad de procesar pedidos o entregas.[4] Los proyectos que exceden el presupuesto son un riesgo común, agravado sin un plan de contingencia.[5] Es como quemar billetes en una hoguera.
- **Disminución de la Eficiencia Operativa:** La incompatibilidad con nuevas tecnologías fuerza procesos manuales y redundantes, ralentizando operaciones y aumentando errores.[4] La deficiente UX y lentitud del sistema reducen la productividad de los empleados, desviándolos de tareas de valor agregado.[4] La productividad se va de vacaciones sin avisar.
- **Riesgos Legales y de Reputación:** El incumplimiento normativo y las vulnerabilidades de seguridad exponen a la empresa a multas, litigios y pérdida de confianza de clientes y socios.[4] La reputación se afecta si los proyectos no cumplen plazos, calidad o presupuesto.[5] Un escándalo en ciernes.
- **Pérdida de Competitividad y Oportunidades de Negocio:** La incapacidad de innovar, integrar funcionalidades o adaptarse al mercado impide a la empresa expandirse, ofrecer servicios diferenciados o competir con empresas más ágiles.[4, 1] Es como intentar ganar una carrera con un triciclo.
- **Insatisfacción del Cliente y Empleado:** La deficiente UX y los errores impactan directamente la satisfacción del cliente, pudiendo llevar a su pérdida.[4] Internamente, la frustración por el sistema afecta la moral de los empleados, contribuyendo al agotamiento y rotación de talento.[5] Nadie quiere trabajar con un sistema que parece sacado de un museo.
- **Dificultad para Atraer y Retener Talento:** La dependencia de tecnologías obsoletas hace la empresa menos atractiva para desarrolladores jóvenes, dificultando la contratación y retención de talento en TI. Los nuevos talentos prefieren ir a empresas con "juguetes" más modernos.
- **Retrasos en Proyectos y Sobrecostos:** La complejidad y fragilidad del sistema legacy hacen que cualquier modificación o adición de funcionalidades sea lenta, costosa y propensa a errores, resultando en retrasos y sobrecostos.[5, 9] Un verdadero "agujero negro" de tiempo y dinero.

Es crucial comprender que los impactos de la inacción se agravan con el tiempo (vulnerabilidades, incompatibilidad, fallos). Esta naturaleza escalonada refuerza la urgencia de

las soluciones, ya que posponer la acción solo incrementa costos y riesgos futuros.

Proponer soluciones

La modernización de un sistema legacy implica un espectro de estrategias, cada una con implicaciones en costo, riesgo y beneficio. Para Logística Ágil S.A., se consideran varios enfoques para una visión integral.

Estrategias y Enfoques de Modernización Considerados

Las principales estrategias para la reingeniería del sistema de Logística Ágil S.A. son:

- **1. Re-hosting (Lift and Shift):**
  - **Descripción:** Implica migrar la aplicación tal cual, con cambios mínimos, de su infraestructura actual (local) a una nueva plataforma, típicamente la nube. Se traslada el software sin modificar su arquitectura interna.
  - **Ventajas:** Estrategia rápida y de menor riesgo inicial, minimizando la intervención en el código. Reduce costos de infraestructura física y mejora la resiliencia y disponibilidad a nivel de plataforma.[6]
  - **Desventajas:** No resuelve los problemas subyacentes del código legacy (deuda técnica, escalabilidad limitada, incompatibilidad). Solo traslada los problemas a un nuevo entorno. Es como cambiar de casa, pero llevarte todos los trastos viejos.
- **2. Re-platforming:**
  - **Descripción:** Consiste en mover la aplicación a una nueva plataforma en la nube, con optimizaciones menores para aprovechar sus capacidades nativas (ej., migrar una base de datos local a un servicio gestionado en la nube).
  - **Ventajas:** Ofrece mejoras en rendimiento, escalabilidad y gestión de infraestructura con menor esfuerzo que una re-arquitectura completa.
  - **Desventajas:** Gran parte del código legacy y sus problemas persisten. Las mejoras en agilidad e innovación son limitadas.
- **3. Re-factoring (Refactorización):**
  - **Descripción:** Implica reestructurar y limpiar el código existente sin cambiar su comportamiento externo, para mejorar su calidad, legibilidad y mantenibilidad. Un ejemplo es refactorizar monolitos a microservicios, extrayendo componentes gradualmente.[7, 8]
  - **Ventajas:** Mejora la mantenibilidad, reduce la deuda técnica y facilita futuras evoluciones. Permite una transición gradual, minimizando el riesgo de interrupciones.[7] Los microservicios pueden implementarse y probarse de forma independiente, flexibilizando las dependencias.[8]
  - **Desventajas:** Puede ser un proceso largo y complejo, especialmente con sistemas monolíticos acoplados. Requiere conocimiento profundo del sistema y puede introducir riesgos si no se gestiona con cuidado.[7]
- **4. Re-architecting (Re-arquitectura):**
  - **Descripción:** Rediseñar y reconstruir la aplicación sustancialmente, adoptando una

arquitectura moderna (microservicios, contenedores, nube). Puede implicar migrar bases de datos jerárquicas a relacionales o NoSQL.[1, 10, 11, 12, 13]
    - **Ventajas:** Permite aprovechar al máximo las tecnologías modernas, logrando escalabilidad, flexibilidad, rendimiento y seguridad óptimos. Aborda la deuda técnica y posiciona a la empresa para la innovación.[1, 10]
    - **Desventajas:** Es la estrategia más costosa, de mayor duración y con mayor riesgo, implicando cambios profundos en tecnología y procesos. Requiere inversión significativa en tiempo, recursos y talento.
- **5. Replace (Reemplazo / Big-Bang):**
    - **Descripción:** Adquirir e implementar un nuevo sistema de software comercial (COTS) o desarrollar una solución nueva desde cero, descartando el sistema legacy.
    - **Ventajas:** Permite adoptar las mejores prácticas de la industria y funcionalidades avanzadas de inmediato. Elimina la necesidad de lidiar con la complejidad del código legacy.
    - **Desventajas:** Estrategia de muy alto riesgo, implicando interrupción significativa del negocio y transición masiva de datos y procesos. El "big-bang" es propenso a fallos catastróficos si no se gestiona con planificación impecable.[11] Es como apretar el botón de autodestrucción y esperar lo mejor.
- **6. Retain (Retener / Mantenimiento Mínimo):**
    - **Descripción:** Mantener el sistema legacy en su estado actual, con mantenimiento mínimo indispensable.
    - **Ventajas:** Costo inicial de inversión cero.
    - **Desventajas:** Agrava todos los problemas identificados (seguridad, costos, incompatibilidad) y aumenta exponencialmente el riesgo a largo plazo. No es una solución sostenible. Es como ponerle una curita a una herida de bala.

La elección de una estrategia de modernización no es única. Cada enfoque equilibra riesgo y recompensa. Decisiones impulsivas sin estrategia clara pueden llevar a soluciones temporales que agravan la situación.[1] Es fundamental alinear la estrategia con los objetivos comerciales, considerando los procesos críticos más impactados.[1]

Solución recomendada

Tras evaluar las estrategias, se propone una solución híbrida para Logística Ágil S.A. que combina re-platforming con una re-arquitectura gradual hacia microservicios. Este enfoque busca mitigar los riesgos de un reemplazo "big-bang" y maximizar el valor de negocio a largo plazo.

Solución Propuesta

La solución propuesta para Logística Ágil S.A. es una migración por fases hacia una arquitectura de microservicios en la nube, usando el patrón "Strangler Fig". Este permite extraer funcionalidades del monolito legacy incrementalmente, reemplazándolas con nuevos microservicios en paralelo hasta que el sistema antiguo sea retirado. Es como ir quitando ladrillos viejos de una pared mientras se ponen nuevos, sin que la pared se caiga.

Nueva Arquitectura y Tecnologías:

La arquitectura futura se basará en principios de computación nativa de la nube (Azure o AWS). Los componentes clave incluirán:

- **Microservicios:** Cada funcionalidad de negocio (ej. gestión de inventarios, procesamiento de pedidos, planificación de rutas) se encapsulará en microservicios independientes, permitiendo su desarrollo, despliegue y escalado autónomo.[7, 8]
- **Bases de Datos Modernas:** Migración de la base de datos jerárquica propietaria a una combinación de bases de datos relacionales (para datos estructurados con integridad referencial) y NoSQL (para cargas de alto volumen, datos dinámicos y escalabilidad horizontal), según las necesidades de cada microservicio.[12, 13]
- **APIs (Interfaces de Programación de Aplicaciones):** Los microservicios se comunicarán entre sí y con sistemas externos a través de APIs bien definidas, facilitando la integración.
- **Contenedores y Orquestación:** Uso de tecnologías como Docker y Kubernetes para empaquetar y gestionar los microservicios, asegurando portabilidad y escalabilidad.[7]
- **Plataformas de Nube:** Aprovechamiento de servicios gestionados de la nube para bases de datos, mensajería, seguridad y monitoreo, reduciendo la carga operativa.

Fases Clave de Implementación:

1. **Evaluación Detallada y Planificación Estratégica:** Se realizará una evaluación exhaustiva del sistema legacy para comprender su construcción, tecnologías, limitaciones, deuda técnica y cuellos de botella.[1] Esto incluye identificar dominios de código cohesionados y de bajo acoplamiento para microservicios.[8] Se establecerán objetivos, métricas de éxito y un plan de migración detallado, incluyendo gestión de riesgos.[1, 6]
2. **Re-platforming Inicial (Migración a la Nube):** Una primera fase implicará migrar el sistema legacy (o partes) a una infraestructura en la nube con cambios mínimos (re-hosting de la aplicación COBOL en VMs o re-platforming de la base de datos a un servicio gestionado). Esta etapa es rápida y ofrece beneficios iniciales de la nube, como

reducción de costos de infraestructura y mejora de la resiliencia.[6]

3. **Extracción Gradual de Microservicios (Patrón Strangler Fig):** Esta es la fase central. Se identificarán funcionalidades críticas o problemáticas del monolito y se reescribirán como nuevos microservicios. El tráfico se redirigirá gradualmente del sistema legacy a los nuevos microservicios. Este proceso incremental minimiza interrupciones y permite probar cada componente antes de su adopción.[7, 8]
4. **Migración de Datos:** A medida que se extraen microservicios, se migrarán los datos relevantes a sus nuevos esquemas. Este paso puede ser costoso y complicado, requiriendo un equipo cualificado.[8] Para bases de datos relacionales a NoSQL, podría ser necesaria la desnormalización de datos para optimizar el nuevo modelo.[12]
5. **Integración y Automatización Continua:** Se establecerán mecanismos robustos para la integración continua (CI) y entrega continua (CD) para los nuevos microservicios, permitiendo despliegues frecuentes y automatizados. También se implementarán herramientas de monitoreo para supervisar el rendimiento y la salud de los nuevos entornos.[7, 8]
6. **Capacitación y Adopción:** Se proporcionará capacitación exhaustiva a usuarios finales y al equipo de TI sobre el uso y mantenimiento del nuevo sistema. La formación es crucial para asegurar una alta tasa de adopción y maximizar la eficiencia.[6]

La elección de un enfoque incremental, en lugar de un reemplazo "big-bang", es fundamental para minimizar el riesgo de interrupciones y permitir una adaptación gradual. La migración de datos, a menudo subestimada, es uno de los aspectos más complejos y críticos, requiriendo planificación meticulosa y experiencia técnica.

Justificación de la Solución

La solución propuesta se justifica por su capacidad para abordar integralmente los problemas del sistema legacy de Logística Ágil S.A., alineando la estrategia tecnológica con los objetivos de negocio.

- **Mitigación de Riesgos:** El enfoque gradual de la re-arquitectura (patrón Strangler Fig) reduce significativamente el riesgo de interrupciones operativas a gran escala, a diferencia de un reemplazo "big-bang".[1] Permite identificar y resolver problemas en pequeñas iteraciones, controlando presupuesto y alcance.[5]
- **Maximización del Valor de Negocio:** La modernización se enfoca en los procesos críticos del negocio, generando un impacto directo en eficiencia y competitividad.[1] Los beneficios esperados incluyen mayor eficiencia operativa, reducción de costos, mejora de transparencia y calidad del producto, y plazos de comercialización más cortos.[2, 3]
- **Eficiencia de Costos a Largo Plazo:** Aunque la inversión inicial en re-arquitectura es considerable, los costos de mantenimiento y operación se reducirán drásticamente a largo plazo. Las tecnologías modernas son más fáciles de mantener, requieren menos expertos en tecnologías obsoletas, y los servicios en la nube optimizan el gasto con modelos de pago por uso.[4, 2, 3]
- **Adaptabilidad y Escalabilidad Futura:** Una arquitectura basada en microservicios y la

nube proporciona la flexibilidad y escalabilidad necesarias para que Logística Ágil S.A. crezca, se adapte y expanda sin las limitaciones actuales.[4, 1, 6] Esto posiciona a la empresa para el éxito a largo plazo.

- **Mejora de la Experiencia de Usuario y Empleado:** Las nuevas interfaces y funcionalidades mejorarán significativamente la productividad y satisfacción de los usuarios finales. La adopción de tecnologías modernas hará a Logística Ágil S.A. más atractiva para el talento tecnológico, facilitando la atracción y retención de desarrolladores.[4]
- **Cumplimiento Normativo y Seguridad Mejorada:** La nueva arquitectura permitirá incorporar nativamente las mejores prácticas de seguridad y cumplir con normativas actuales, reduciendo la exposición a riesgos legales y de reputación.[4, 2, 3]

Este enfoque equilibrado permite a la empresa obtener beneficios incrementales mientras gestiona los riesgos del proyecto. La solución no solo "arregla" el sistema obsoleto, sino que transforma la capacidad tecnológica de la empresa, permitiéndole innovar y competir eficazmente.

Conclusión

La reingeniería de un sistema legacy, como el caso hipotético de Logística Ágil S.A., es un desafío formidable pero ineludible. Este estudio me ha permitido analizar la interacción de factores técnicos, operativos y organizacionales que llevan a la obsolescencia y la necesidad de modernización, con el análisis nos muestra que los problemas de los sistemas heredados (seguridad, costos, incompatibilidad, escalabilidad) no son aislados, sino una red de deficiencias que se exacerban, impactando eficiencia, competitividad e innovación. Su persistencia se debe a la deuda técnica y la resistencia al cambio, lo que indica que la modernización es un reto tecnológico y de gestión cultural, la solución propuesta, una re-arquitectura progresiva hacia microservicios en la nube usando el patrón "Strangler Fig", es una senda viable y estratégica. Este enfoque permite una transición controlada, minimizando riesgos y distribuyendo la complejidad en fases manejables. Al adoptar una arquitectura moderna, asi mismo Logística Ágil S.A. resolverá sus problemas y se posicionará para el crecimiento futuro, mejorando agilidad, escalabilidad e integración de nuevas tecnologías, la reingeniería de software es un proceso continuo de adaptación y mejora, que curiosamente el éxito depende de la tecnología adecuada, la planificación meticulosa, gestión de riesgos proactiva, comunicación efectiva y un liderazgo comprometido con el cambio organizacional.

Referencias

- [4]https://www.captia.es/blog/aplicaciones-legacy.html
- [1]https://www.codurance.com/es/publications/modernizacion-sistemas-heredados-retos-soluciones
- [10]https://www.orbit.es/que-hacer-con-los-sistemas-legacy-o-heredados/
- [11]https://arsum.cl/?page_id=588
- [2]https://www.ibm.com/es-es/think/topics/business-process-reengineering-examples
- [3]https://www.ibm.com/mx-es/think/topics/business-process-reengineering-examples
- [1]https://www.codurance.com/es/publications/modernizacion-sistemas-heredados-retos-soluciones
- [6]https://www.door3.com/es/blog/legacy-system-modernization-approaches-to-improve-software
- [7]https://itequia.com/es/migrar-de-monolitos-a-microservicios-cuatro-estrategias-exitosas/
- [8]https://www.sysviewsoft.com/portal/index.php/servicios/plataforma-open/migracion-de-aplicaciones-hacia-microservicios
- [12]https://learn.microsoft.com/es-es/azure/cosmos-db/nosql/migrate-relational-data
- [13]https://learn.microsoft.com/es-es/dotnet/architecture/cloud-native/relational-vs-nosql-data
- [5]https://www.startechup.com/es/blog/10-common-software-development-risks/
- [9]https://www.lucidchart.com/blog/es/retos-del-desarrollo-de-software
- Trello:https://trello.com/invite/b/68a2af82cda31dd11f40f98f/ATTId6e7b012b51244152dbc722be95aa5b5176F9DBA/reingenieria-de-un-sistema-legacy
- Github:https://github.com/Noobli11/Miguel-Motta