Cambio de Arquitectura Monolítica a Arquitectura basada en Microservicios y Serverless para el aplicativo inmobiliario Mobilia

Diego Alejandro Poveda Alzate

Informe de prácticas académicas para optar al título de Ingeniero de Sistemas

Asesor interno

Sandra Patricia Zabala Orrego, Ingeniera informática, Especialista en Gerencia

Asesor externo

Jorge Luis Gutiérrez Moncada, Administrador de Sistemas Informáticos, Especialista en Business Intelligence

Universidad de Antioquia

Facultad de Ingeniería

Ingeniería de Sistemas

Medellín

2024

TABLA DE CONTENIDO

LISTA DE FIGURAS

SIGLAS, ACRÓNIMOS Y ABREVIATURAS

UdeA	Universidad de Antioquia
CI	Continuous Integration (Integración Continua)
CD	Continuous Delivery (Entrega Continua)
Fig	Figura
S.A.S	Sociedad por Acciones Simplificada
QA	Quality Assurance (Analistas de Aseguramiento de la Calidad)

RESUMEN

Mobilia Software, una compañía originada de la inmobiliaria Acrecer S.A.S., ofrece un sistema de software inmobiliario llamado "Mobilia" que ha alcanzado sus límites operativos debido a su arquitectura monolítica. Para mejorar la escalabilidad, flexibilidad y eficiencia operativa, se ha propuesto migrar esta arquitectura a un modelo basado en microservicios y serverless. Este cambio permite descomponer el sistema en servicios más pequeños y cohesivos, mejorando la capacidad de implementación continua y la adopción de nuevas tecnologías. El proyecto se desarrolló en varias fases, comenzando con la evaluación y documentación del sistema actual, seguido del diseño e implementación de la nueva arquitectura, y culminando con una estrategia de migración y despliegue. Utilizando metodologías ágiles y prácticas de ingeniería de software, se buscó minimizar los riesgos y maximizar los beneficios de esta transición. A pesar de retos tales como la coordinación con una consultora externa, o la capacitación del equipo de desarrollo en nuevas tecnologías y la gestión de errores en la aplicación base, se espera que la migración resulte en un sistema más adaptable y escalable, alineado con las demandas del mercado y las expectativas de los usuarios.

Palabras clave — **Migración de arquitectura, microservicios, serverless, arquitectura monolítica, escalabilidad, flexibilidad.**

ABSTRACT

Mobilia Software, a company originated from the real estate firm Acrecer S.A.S., offers a real estate software system called "Mobilia" that has reached its operational limits due to its monolithic architecture. To improve scalability, flexibility, and operational efficiency, a migration to a microservices and serverless architecture has been proposed. This change allows for the decomposition of the system into smaller, cohesive services, enhancing continuous deployment capabilities and the adoption of new technologies. The project was developed in several phases, starting with the evaluation and documentation of the current system, followed by the design and implementation of the new architecture, and culminating with a migration and deployment strategy. Using agile methodologies and software engineering practices, the aim was to minimize risks and maximize the benefits of this transition. Despite challenges such as coordination with an external consultancy, training the development team in new technologies, and managing errors in the base application, the migration is expected to result in a more adaptable and scalable system, aligned with market demands and user expectations.

Keywords — **Architecture migration, microservices, serverless, monolithic architecture, scalability, flexibility.**

I. INTRODUCCIÓN

La compañía Mobilia Software, nacida a partir de la empresa inmobiliaria Acrecer S.A.S, ofrece un servicio de software inmobiliario llamado “Mobilia”, para que las inmobiliarias interesadas puedan gestionar sus inmuebles, contratos, pagos y facturación en un solo lugar.

Este proyecto propone la migración del software, el cual está realizado bajo una arquitectura monolítica, a una arquitectura basada en microservicios y serverless, con el objetivo de aumentar la escalabilidad, mejorar la flexibilidad y optimizar la eficiencia operativa del sistema de este software.

La arquitectura monolítica, caracterizada por su diseño unificado, aunque funcional para etapas iniciales o proyectos de menor envergadura, plantea significativas limitaciones en términos de escalabilidad, mantenimiento y velocidad de implementación de nuevas características a medida que el sistema crece **[1]**. Estas limitaciones pueden resultar en cuellos de botella operativos, dificultades para adaptarse a nuevas tecnologías y un mayor tiempo de lanzamiento al mercado de nuevas funcionalidades.

Frente a estos desafíos, la arquitectura de microservicios emerge como una solución que permite descomponer el sistema en servicios más pequeños, independientes y cohesivos, cada uno de los cuales realiza una función específica. Este enfoque promueve una mayor agilidad y escalabilidad, facilitando el desarrollo, despliegue y mantenimiento de cada servicio de manera independiente. Además, la incorporación del modelo serverless en este paradigma arquitectónico ofrece un nivel adicional de optimización, al abstraer la gestión de la infraestructura y permitir que el equipo de desarrollo se enfoque en la creación de valor a través del código, mejorando así la eficiencia operativa y reduciendo costos asociados a la gestión de servidores.

El proyecto se estructura en varias fases, desde la planificación y análisis del sistema actual, pasando por el diseño e implementación de la nueva arquitectura, hasta la estrategia de migración, despliegue y operación continua. Se adoptará una metodología que integra prácticas de desarrollo ágil, estrategias específicas para la implementación de microservicios y principios de ingeniería de software, buscando minimizar los riesgos y maximizar los beneficios de esta transición.

II. PLANTEAMIENTO DEL PROBLEMA

Mobilia Software, desarrollado por Acrecer S.A.S., actualmente enfrenta limitaciones significativas en términos de escalabilidad, mantenimiento, flexibilidad y costos operativos debido a su arquitectura monolítica. Con el crecimiento continuo del número de usuarios y la complejidad del sistema, se han presentado cuellos de botella que afectan el rendimiento y la capacidad de respuesta del sistema, además de tiempos de inactividad y mayores costos de mantenimiento. La estructura monolítica dificulta la implementación de nuevas funcionalidades y la adopción de nuevas tecnologías, limitando la capacidad del sistema para evolucionar y adaptarse a las demandas del mercado **[2]**. Para abordar estos desafíos y asegurar la continuidad y eficiencia operativa, se propone migrar a una arquitectura basada en microservicios y serverless, permitiendo una mayor escalabilidad, flexibilidad y una optimización de recursos que reducirá los costos operativos y mejorará la experiencia del usuario **[3]**.

III. JUSTIFICACIÓN

La migración del software inmobiliario Mobilia de una arquitectura monolítica a una arquitectura basada en microservicios y serverless es una respuesta necesaria a las limitaciones estructurales y operativas que el sistema actual enfrenta en un entorno tecnológico y de negocio en constante evolución. La arquitectura monolítica, aunque adecuada en las primeras etapas de desarrollo por su simplicidad y facilidad de manejo, ha mostrado ser insuficiente para satisfacer las demandas de escalabilidad y flexibilidad requeridas por un software que debe atender un creciente número de usuarios y funcionalidades cada vez más complejas. Los cuellos de botella operativos, junto con los elevados costos de mantenimiento y los tiempos de inactividad, son claros indicios de que la infraestructura actual no está preparada para sostener el crecimiento proyectado ni para adaptarse con agilidad a las nuevas exigencias del mercado **[3]**.

En contraste, la arquitectura de microservicios ofrece una solución moderna que descompone el sistema en módulos más pequeños, independientes y altamente cohesivos. Esta descomposición permite una mayor agilidad en el desarrollo, pruebas y despliegue de nuevas funcionalidades, lo cual es crucial en un entorno donde la rapidez en la respuesta al mercado es determinante para el éxito. Además, la integración de un modelo serverless en este nuevo paradigma arquitectónico no solo mejora la flexibilidad y escalabilidad del sistema, sino que también optimiza la gestión de recursos, permitiendo que el equipo de desarrollo se enfoque en la creación de valor a través del código, en lugar de en la administración de la infraestructura. Esto no solo reduce los costos operativos, sino que también mejora la experiencia del usuario al asegurar un sistema más robusto y con mayor capacidad de respuesta **[4]**.

Por lo tanto, la migración a una arquitectura de microservicios y serverless es esencial para asegurar que Mobilia continúe siendo competitiva, capaz de escalar según sea necesario y de adaptarse rápidamente a nuevas tecnologías y demandas del mercado. Esta transformación permitirá a Mobilia superar las limitaciones de su arquitectura actual y posicionarse mejor para el crecimiento y la innovación futuros.

IV. OBJETIVOS

A. Objetivo general

Planificar, diseñar e implementar una arquitectura basada en microservicios y serverless para mejorar la escalabilidad, flexibilidad, y eficiencia operativa del sistema actual, asegurando la continuidad del negocio y una mejor experiencia de usuario.

B. Objetivos específicos

- Evaluar y documentar las funcionalidades, procesos, y limitaciones del sistema monolítico actual para determinar los componentes candidatos a ser migrados a microservicios y serverless.
- Identificar y aplicar soluciones serverless para aquellas operaciones que pueden beneficiarse de la gestión automática de recursos y escalabilidad, reduciendo los costos operativos y mejorando la eficiencia.
- Desarrollar e implementar cada microservicio según el diseño propuesto, asegurando la compatibilidad y la comunicación efectiva entre servicios mediante APIs y eventos.

V. MARCO TEÓRICO

El cambio hacia arquitecturas basadas en microservicios y serverless representa no solo una evolución tecnológica, sino también un cambio en la cultura de desarrollo de software, impulsando la adopción de prácticas ágiles, CI y CD, para facilitar el desarrollo, el despliegue y el mantenimiento de aplicaciones complejas **[5]**.

Este marco teórico investiga los principios básicos, las ventajas, los retos y las estrategias para implementar la arquitectura de microservicios y la computación serverless. Se examinan en detalle los principios de diseño, las tecnologías pertinentes, las metodologías de gestión de proyectos de software y las tácticas de migración. Esto proporciona un entendimiento exhaustivo de cómo estas innovaciones pueden revolucionar el desarrollo de software y alinear más eficazmente la tecnología con los objetivos empresariales. Mediante la revisión de literatura y análisis de casos prácticos, este estudio ofrece una visión completa sobre las mejores prácticas y las lecciones aprendidas durante el proceso de transición de sistemas monolíticos a arquitecturas más adaptables y escalables.

La arquitectura monolítica se caracteriza por agrupar todos los componentes de una aplicación en un único bloque de software cohesionado. En este modelo, la interfaz de usuario, la lógica de negocio y el acceso a la base de datos están interrelacionados, funcionando como una única entidad que se compila y despliega en conjunto. Este tipo de arquitectura es sencillo de desarrollar y gestionar en las primeras etapas de un proyecto debido a su estructura unificada, lo que permite una rápida implementación inicial. Sin embargo, a medida que el sistema crece en tamaño y complejidad, comienzan a surgir limitaciones importantes. La escalabilidad se ve comprometida ya que cualquier cambio en una parte del sistema implica recompilar y desplegar la totalidad del software, lo que aumenta el riesgo de errores e interrupciones en la operación.

Además, la arquitectura monolítica no favorece la incorporación ágil de nuevas funcionalidades ni la adopción de nuevas tecnologías. La interdependencia de los módulos genera cuellos de botella, dificultades en el mantenimiento y una mayor complejidad para el trabajo colaborativo en grandes equipos de desarrollo. Por estas razones, las empresas que desean mejorar su flexibilidad y capacidad de respuesta ante cambios tecnológicos y del mercado optan por migrar a arquitecturas más modernas, como la de microservicios.

La arquitectura de microservicios se plantea como una solución a los problemas de escalabilidad y flexibilidad que enfrenta una arquitectura monolítica. Este enfoque promueve la descomposición del sistema en servicios independientes y más pequeños, donde cada microservicio realiza una función específica dentro del sistema global. Estos microservicios se comunican entre sí a través de interfaces bien definidas, generalmente utilizando APIs o mensajes de eventos.

Una de las principales ventajas de los microservicios es la independencia en el ciclo de desarrollo, despliegue y mantenimiento. Cada microservicio puede ser desarrollado y desplegado de manera autónoma, sin afectar a los demás componentes del sistema. Esto permite una mayor agilidad en el desarrollo de nuevas características y en la corrección de errores, así como la posibilidad de escalar solo aquellos microservicios que lo necesiten, en lugar de todo el sistema. Además, al ser independientes, los microservicios pueden desarrollarse utilizando diferentes tecnologías, lo que facilita la adopción de nuevas herramientas y lenguajes de programación.

Sin embargo, la implementación de microservicios no está exenta de desafíos. La gestión de la comunicación entre servicios, el monitoreo de múltiples instancias y la implementación de estrategias de tolerancia a fallos son algunos de los aspectos que requieren atención adicional en comparación con un sistema monolítico. Aun así, los beneficios a largo plazo en términos de escalabilidad y mantenimiento hacen de los microservicios una opción atractiva para sistemas en crecimiento.

VI. METODOLOGÍA

El proyecto se desarrolla haciendo uso SCRUM, el cual es un marco de trabajo ágil para la gestión y desarrollo de proyectos complejos, que se caracteriza por su enfoque iterativo e incremental. Esta metodología se centra en entregar productos de alta calidad mediante la colaboración continua, la mejora incremental y la adaptación rápida a los cambios.

El proceso de SCRUM se apoya en la colaboración constante entre los miembros del equipo, la transparencia en la comunicación y la capacidad de adaptación rápida a los cambios **[6]**.

En SCRUM, las ceremonias o eventos son esenciales para asegurar que el equipo trabaje de manera alineada y eficiente, manteniendo el enfoque en la entrega de valor en cada iteración.

A. Daily Scrum

La reunión diaria es un evento corto en el que todo el equipo de desarrollo se reúne para sincronizar el trabajo y planificar el día **[6]**. Durante esta reunión, cada miembro del equipo responde tres preguntas clave: ¿Qué hice ayer?, ¿Qué planeo hacer hoy?, ¿Tengo algún impedimento que me impida avanzar?

El objetivo principal de esta reunión es proporcionar visibilidad sobre el estado del trabajo, identificar posibles bloqueos y ajustar el plan de trabajo diario. Es importante que esta reunión sea breve y directa para mantener el enfoque en el progreso del sprint.

B. Sprint Planning

La Spring Planning es la reunión donde el equipo de desarrollo, junto con el Product Owner, define qué trabajo se realizará en el próximo sprint **[6]**. Durante esta ceremonia, el equipo selecciona las historias de usuario o tareas que se encuentran en el backlog del producto y las traslada al backlog del sprint. Para decidir qué ítems se incluirán, el equipo debe considerar la prioridad establecida por el Product Owner y la capacidad de trabajo disponible para el sprint.

El equipo también define el "Objetivo del Sprint", que es una meta clara y alcanzable que guiará el desarrollo durante el sprint. El resultado de esta reunión es un compromiso por parte del equipo para completar el trabajo acordado dentro del tiempo asignado.

C. Sprint Review

La Sprint Review se realiza al final de cada sprint y tiene como propósito inspeccionar el incremento de producto desarrollado durante ese período. En esta ceremonia, el equipo de desarrollo presenta el trabajo completado al Product Owner y a los stakeholders **[6]**. Se evalúa si los entregables cumplen con la definición de "hecho" y se demuestra cómo las nuevas funcionalidades funcionan en el sistema.

Durante la Review, el equipo recibe retroalimentación por parte del Product Owner y los stakeholders, lo que puede influir en los próximos sprints. Esta ceremonia es crucial para ajustar el backlog y asegurar que el producto continúe alineado con las expectativas del negocio y los usuarios.

A partir del software inmobiliario existente, se realizaron estas ceremonias para la planeación, diseño e implementación de la nueva arquitectura del sistema, el cual se realiza de manera iterativa y modularizada; es decir, se identifican los módulos críticos del sistema actual, los cuales son los que más se beneficiarían de implementarse en microservicios, y a partir de este análisis se define cual módulo del software inmobiliario es el siguiente a implementar.

VII. RESULTADOS

A. Sistema de Mobilia

Como se puede entender en base a la Fig. 1, el aplicativo Mobilia, a pesar de que continúa en constante planeación, diseño e implementación de su modularización en microservicios, sigue funcionando como una aplicación monolítica, donde todos los componentes de la aplicación, como la interfaz de usuario, la lógica de negocio, y el almacenamiento de datos, se encuentran integrados en un solo código base y se gestionan como una única unidad. Esto significa que cualquier cambio en una parte del sistema requiere la compilación y el despliegue del sistema completo **[4]**.

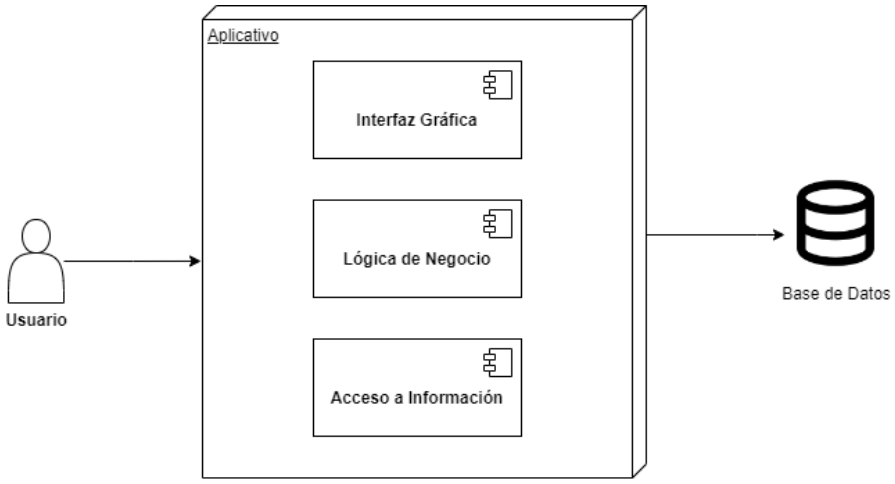

Fig. 1. Arquitectura monolítica

Además de distintas funcionalidades relacionadas a la lógica de negocio del sistema, también es necesario adaptar otras funcionalidades a arquitectura de microservicios, como lo son los CRON Jobs, que se ejecutan cada cierta cantidad de tiempo, realizando una acción que se les fue asignada.

B. Módulo de Reportes

El módulo de reportes de Mobilia es constituido por un backend realizado en arquitectura de microservicios, haciendo uso del framework Spring Boot para Java, el cual recibe peticiones enviadas desde un frontend realizado con Next.js, un framework el cual permite realizar aplicaciones escalables y mantenibles haciendo uso de la librería de ReactJS.

El módulo de reportes se encarga únicamente de operaciones relacionadas con el área, es decir, no realiza acciones pertinentes a otros módulos, lo cual es importante para seguir la norma de que los microservicios deben ser independientes, y no abordar temas que no le pertenezcan al microservicio, tal como se observa en la Fig. 2.

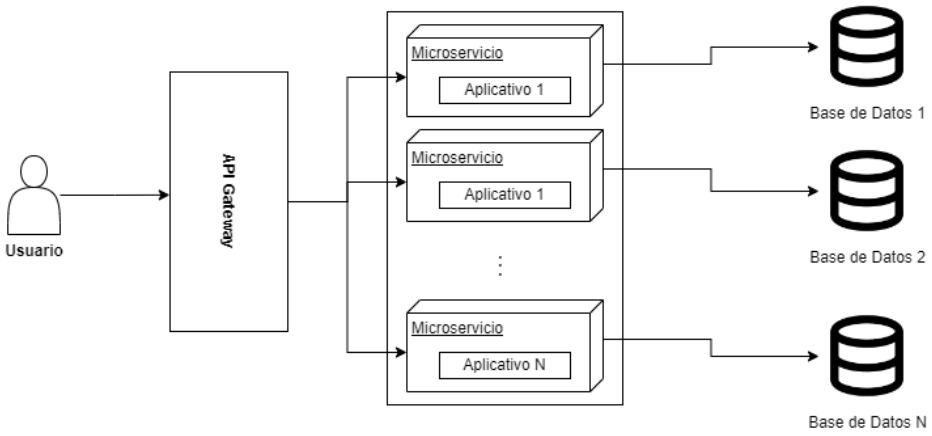

Fig. 2. Arquitectura de Microservicios

C. Integración de ambos sistemas

La manera en que ambos sistemas interactúan entre si es de vital importancia, ya que es necesario que los clientes del aplicativo tengan una experiencia lo más cómoda e intuitiva posible, por lo cual es necesario que se pueda navegar entre un sistema y el otro sin muchas complicaciones.

Para lograr este comportamiento, se decidió que cada vez que un cliente decida ingresar el módulo de reportes, Mobilia será el encargado de realizar el cambio de aplicativo, permitiendo que el cliente no deba preocuparse por ingresar manualmente la URL de un aplicativo o el otro.

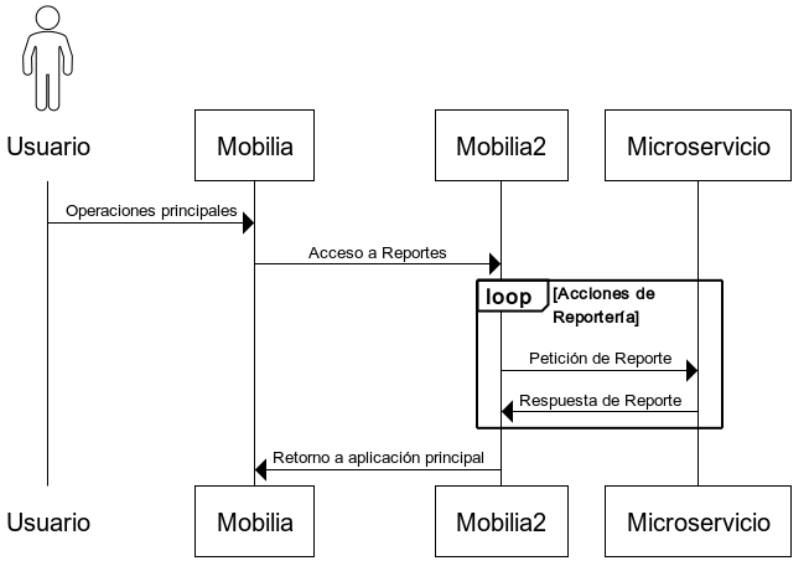

Fig. 3. Interacción entre Mobilia Actual y el Módulo de Reportes

Como se observa en la Fig. 3, un usuario de Mobilia empieza sus operaciones en el sistema actual, por lo que este, inicialmente, no tiene conocimiento de la otra instancia de Mobilia que aloja el módulo de Reportes. Una vez un usuario con acceso a reportes decide realizar una acción pertinente al módulo, el sistema de Mobilia actual realiza una redirección al nuevo sistema, enviando información como la sesión del usuario y la pantalla a la cual accedió, para que todo el proceso sea fluido y no necesite interacciones extras por parte del usuario.

Una vez en el módulo de Reportes, el usuario crea o ejecute tantos reportes como sea necesario, de forma que no es necesario interactuar con el sistema anterior.

Cuando el usuario decide retornar al aplicativo inicial, el módulo de Reportes le facilita la opción de volver al Mobilia inicial, donde el usuario podrá continuar sus operaciones como lo hacía anteriormente.

D. Impedimentos y dificultades presentadas

Durante la fase de migración a la nueva arquitectura de microservicios, se lograron avances significativos, como el desarrollo e implementación del módulo de reportes bajo la nueva interfaz de usuario. Sin embargo, se presentaron varios desafíos que impactaron el cronograma del proyecto y la eficiencia del desarrollo.

En primer lugar, se detectaron múltiples errores en la aplicación base que causaron demoras importantes en la integración del módulo de reportes con el microservicio correspondiente. Para abordar esta situación, se estableció un equipo dedicado a la identificación, documentación y resolución de estos bugs. Este equipo, compuesto por desarrolladores y QAs, trabaja de manera

sistemática para registrar y gestionar los errores, lo que permite avanzar de manera más organizada y eficaz en la solución de los problemas.

Además, la dependencia de la consultora externa generó retrasos debido a dificultades para coordinar reuniones y alinear los horarios de trabajo entre ambas partes. Esta falta de sincronización afecta el ritmo de la migración, ya que en varias ocasiones se han tenido que posponer actividades clave. Para mitigar este riesgo, se implementó un calendario de reuniones recurrentes y el uso de herramientas de gestión compartida que faciliten la planificación y ejecución conjunta del proyecto.

A pesar de estos obstáculos, los avances en la migración y la integración de los servicios continúan, con un enfoque en la mejora de la comunicación entre los equipos y la resolución eficiente de los errores detectados.

VIII. CONCLUSIONES

La transición de una arquitectura monolítica a una basada en microservicios y serverless proporciona mejoras significativas en la escalabilidad y flexibilidad del sistema Mobilia. Este enfoque permite un desarrollo más modular e independiente, optimiza la eficiencia operativa y facilita la incorporación de nuevas tecnologías y funcionalidades.

Es fundamental optimizar la integración entre los diversos módulos y microservicios del sistema, priorizando una gestión eficiente de errores y una coordinación fluida con la consultora externa, para garantizar un funcionamiento cohesivo y minimizar interrupciones en el servicio.

Es importante invertir en la capacitación continua del equipo de desarrollo para asegurar una transición fluida hacia la nueva arquitectura y maximizar los beneficios a largo plazo del sistema actualizado.

La modularidad alcanzada con la implementación de una arquitectura de microservicios facilita la colaboración entre el equipo de desarrollo, permitiendo una mejor asignación de responsabilidades y promoviendo una mayor independencia en la implementación de nuevas características.

Fortalecer la observabilidad del sistema mediante la implementación de herramientas de monitoreo y trazabilidad, permite detectar y resolver problemas en tiempo real, asegurando así un rendimiento óptimo y la continuidad del servicio.

REFERENCIAS

[1] O. Al-Debagy, and P. Martinek. “A Comparative Review of Microservices and Monolithic Architectures” [En línea]. Disponible en: https://arxiv.org/pdf/1905.07997.

[2] J. P. Gouigoux, D. Tamzalit. “From Monolith to Microservices: Lessons Learned on an Industrial Migration to a Web Oriented Architecture” [En línea]. Disponible en: https://hal.science/hal-01873948/document.

[3] M. Kohlberg. “Monolithic Architecture: The Basics and Beyond”. [En línea]. Disponible en: https://encore.dev/resources/monolithic-architecture

[4] Open Oregon State. “Monolith versus Microservice Architectures – Handbook of Software Engineering Methods” [En línea]. Disponible en: https://open.oregonstate.education/setextbook/chapter/monolith-versus/

[5] M. Seedat, Q. Abbas, N. Ahmad. “Systematic Mapping of Monolithic Applications to Microservices Architecture” [En línea]. Disponible en: https://ar5iv.labs.arxiv.org/html/2309.03796

[6] S. Sachdeva. “Scrum Methodology” [En línea]. Disponible en: https://bit.ly/4e6rKwE