Revista Multidisciplinaria RIIDG
ISSN: 3091-194X
Doi: https://doi.org/10.64041/riidg.v3i1.20

Revista Internacional de Investigación y Desarrollo Global

Periocidad trimestral, Volumen 3, Numero 1, Años (2024), Pag. 31-45

Propuesta de modernización arquitectónica de sistemas de gestión en telefonía móvil mediante microservicios.

Proposal for the architectural modernization of mobile phone management systems through microservices

AUTORES

Yumi Taday Jonathan Alexander
Universidad Agraria del Ecuador
Ecuador – Milagro
Jonathan.yumi.taday@uagraria.edu.ec
https://orcid.org/0009-0003-7167-5363

Como citar:	Fecha de recepción:20224-01-15
Yumi Taday, J. A. (2024). Propuesta de modernización arquitectónica de sistemas de gestión en telefonía móvil mediante microservicios. *Revista Internacional De Investigación Y Desarrollo Global*, *3*(1), 31–45.	Fecha de aceptación: 2024-02-15 Fecha de publicación: 2024-03-15

Revista Multidisciplinaria RIIDG
ISSN: 3091-194X
Doi: https://doi.org/10.64041/riidg.v3i1.20

Revista Internacional de Investigación y Desarrollo Global

Resumen

La propuesta de rediseño arquitectónico plantea la migración de un sistema monolítico de gestión de informes y recomendaciones en el sector de telefonía móvil hacia una arquitectura distribuida basada en microservicios, con el objetivo de mejorar la escalabilidad, mantenibilidad, resiliencia y despliegue continuo del sistema, respondiendo a las crecientes demandas del entorno tecnológico y del negocio digital. Este rediseño parte de un análisis exhaustivo del sistema actual, cuya estructura monolítica ha demostrado limitaciones significativas debido a su rigidez, acoplamiento excesivo entre módulos, baja tolerancia a fallos, tiempos prolongados de respuesta ante incidentes, y dificultad para introducir nuevas funcionalidades sin comprometer la estabilidad del sistema completo. En este contexto, se propone la adopción del patrón de migración Strangler, que permite una transición progresiva mediante la segmentación del sistema en dominios funcionales bien definidos, como autenticación, generación de informes, análisis de datos, y gestión de usuarios, los cuales se desarrollan como microservicios autónomos, desacoplados y desplegables de forma independiente. Para ello, se integran tecnologías modernas que soportan la ejecución distribuida y la escalabilidad horizontal, tales como contenedores Docker, orquestación con Kubernetes, interfaces API REST, y sistemas de mensajería asincrónica como Apache Kafka o RabbitMQ, que facilitan la comunicación eficiente entre servicios. Adicionalmente, se incorporan servicios transversales esenciales como observabilidad (logeo, métricas y trazabilidad distribuida), monitoreo en tiempo real (con Prometheus y Grafana), gestión centralizada de configuración, y mecanismos de seguridad robusta basados en protocolos de autenticación como OAuth2. La propuesta se enmarca dentro de una cultura DevOps, promoviendo la integración y entrega continua (CI/CD) mediante herramientas como Jenkins, GitLab CI o GitHub Actions, lo que permite automatizar pruebas, validaciones y despliegues, minimizando errores y optimizando los ciclos de desarrollo. Entre los beneficios esperados de esta transformación destacan la escalabilidad horizontal de cada servicio de forma independiente, mayor tolerancia a fallos, despliegues más ágiles y seguros, así como una notable flexibilidad tecnológica que permite la evolución constante del sistema sin dependencia del stack tecnológico original. No obstante, se reconoce que esta arquitectura distribuida conlleva nuevos desafíos, como la complejidad en la gestión de transacciones distribuidas, la necesidad de una trazabilidad avanzada para el diagnóstico de errores, y una orquestación eficiente de servicios para garantizar la cohesión y el rendimiento global del sistema. En conjunto, esta propuesta representa un avance sustancial hacia la modernización de plataformas críticas en el sector de telecomunicaciones, alineada con principios de arquitectura moderna, ingeniería de software y transformación digital sostenible.

Palabras clave: Microservicios, Arquitectura distribuida, Análisis de datos, API REST, Monitoreo

Revista Multidisciplinaria RIIDG
ISSN: 3091-194X
Doi: https://doi.org/10.64041/riidg.v3i1.20

Revista Internacional de Investigación y Desarrollo Global

Abstract

The architectural redesign proposal aims to migrate a monolithic system for managing reports and recommendations in the mobile telecommunications sector toward a distributed architecture based on microservices, with the goal of improving the system's scalability, maintainability, resilience, and continuous deployment, in response to the increasing demands of the technological environment and the digital business landscape. This redesign is based on a thorough analysis of the current system, whose monolithic structure has shown significant limitations due to its rigidity, excessive module coupling, low fault tolerance, long response times to incidents, and difficulties in introducing new functionalities without compromising the stability of the entire system. In this context, the adoption of the Strangler migration pattern is proposed, enabling a progressive transition by segmenting the system into well-defined functional domains such as authentication, report generation, data analysis, and user management, each developed as independent, decoupled, and individually deployable microservices. To support this transition, modern technologies that facilitate distributed execution and horizontal scalability are integrated, including Docker containers**,** Kubernetes orchestration**,** REST APIs**,** and asynchronous messaging systems such as Apache Kafka or RabbitMQ, which ensure efficient communication between services. Additionally, essential cross-cutting services are incorporated, such as observability (logging, metrics, and distributed tracing), real-time monitoring (with Prometheus and Grafana), centralized configuration management**,** and robust security mechanisms based on authentication protocols like OAuth2**.** The proposal is framed within a DevOps culture, promoting continuous integration and deployment (CI/CD) through tools such as Jenkins, GitLab CI, or GitHub Actions, enabling the automation of testing, validation, and deployment processes, reducing errors and optimizing development cycles. Among the expected benefits of this transformation are the horizontal scalability of each service independently, increased fault tolerance, more agile and secure deployments**,** and significant technological flexibility that allows the system to evolve continuously without dependence on the original technology stack. However, it is acknowledged that this distributed architecture brings new challenges, such as the complexity of managing distributed transactions, the need for advanced traceability for error diagnosis, and the efficient orchestration of services to ensure overall system cohesion and performance. Overall, this proposal represents a significant step toward the modernization of critical platforms in the telecommunications sector, aligned with the principles of modern architecture, software engineering, and sustainable digital transformation.

Keywords: Microservices, Distributed architecture, Data analysis, REST API, Monitoring.

Revista Multidisciplinaria RIIDG
ISSN: 3091-194X
Doi: https://doi.org/10.64041/riidg.v3i1.20

Introducción

En el contexto actual de transformación digital, las organizaciones del sector de las telecomunicaciones enfrentan la necesidad urgente de adaptar sus sistemas tecnológicos a modelos más escalables, flexibles y resilientes. Uno de los principales desafíos que enfrentan estas empresas es la dependencia de arquitecturas monolíticas tradicionales, las cuales, si bien fueron adecuadas en etapas iniciales de desarrollo, hoy representan un obstáculo para el crecimiento, la innovación y la capacidad de respuesta ante cambios en el entorno tecnológico y comercial (Newman, 2015)

Los sistemas monolíticos se caracterizan por agrupar en una sola aplicación todos los componentes funcionales, lo que incrementa la complejidad en el mantenimiento, dificulta la implementación de nuevas funcionalidades y limita la escalabilidad horizontal (Richardson, 2018). En un sector dinámico como el de la telefonía móvil, donde la generación de informes en tiempo real y la entrega de recomendaciones personalizadas son fundamentales para la toma de decisiones y la experiencia del cliente, estas limitaciones se traducen en pérdidas de competitividad y eficiencia operativa (Fowler, 2019)

Frente a este panorama, surge como alternativa la adopción de una arquitectura de microservicios, entendida como una estrategia de diseño en la que las aplicaciones se construyen como un conjunto de servicios pequeños, independientes y desplegables de forma autónoma, que colaboran entre sí a través de interfaces ligeras, normalmente utilizando APIs REST (Dragoni, Lanese, et al., 2017). Esta arquitectura permite una mayor capacidad de escalamiento, facilita la integración continua, mejora la tolerancia a fallos y optimiza el uso de recursos, aspectos cruciales en entornos de alta demanda como el de las telecomunicaciones móviles(Bass et al., 2015)

En este sentido, el presente trabajo propone un rediseño arquitectónico orientado a migrar un sistema monolítico de gestión de informes y recomendaciones en el sector de telefonía móvil hacia una arquitectura distribuida basada en microservicios. Para ello, se plantea una estrategia de migración progresiva utilizando el patrón Strangler, permitiendo la transición modular y controlada del sistema, mitigando riesgos y garantizando la continuidad del servicio (M. Fowler et al., 2004). Asimismo, se incorporan tecnologías actuales como contenedores Docker, orquestación con Kubernetes, mensajería asincrónica con Apache Kafka y servicios transversales para monitoreo, trazabilidad y seguridad.

Con este enfoque, se busca responder a las crecientes exigencias del sector, proporcionando una infraestructura tecnológica moderna, escalable y orientada a servicios, que permita a las organizaciones de telecomunicaciones mejorar su eficiencia, adaptabilidad y sostenibilidad tecnológica en un entorno cada vez más competitivo.

Revista Multidisciplinaria RIIDG
ISSN: 3091-194X
Doi: https://doi.org/10.64041/riidg.v3i1.20

Material y métodos

Material

Para llevar a cabo el proceso de rediseño arquitectónico y migración del sistema monolítico hacia una arquitectura basada en microservicios, se utilizaron los siguientes recursos tecnológicos y herramientas:

Lenguajes y frameworks

- Java con Spring Boot para microservicios principales.
- Node.js con Express.js para servicios ligeros y de comunicación.
- Python para procesamiento de datos y servicios analíticos.

Contenerización y orquestación

- Docker para la creación de contenedores.
- Kubernetes para la orquestación, escalado y despliegue automatizado.

Comunicación y mensajería

- API REST para comunicación síncrona entre microservicios.
- Apache Kafka y RabbitMQ para mensajería asincrónica y desacoplamiento.

Base de datos

- PostgreSQL para almacenamiento relacional.
- MongoDB para servicios que requieren bases de datos NoSQL.

Monitoreo y observabilidad

- Prometheus y Grafana para métricas y visualización.
- Elastic Stack (ELK) para trazabilidad y análisis de logs.

Seguridad

- OAuth2 y Keycloak para autenticación y autorización centralizada.
- JWT para control de acceso en servicios desacoplados.

Automatización de procesos DevOps

- GitHub Actions, GitLab CI y Jenkins para integración y entrega continua (CI/CD).
- Swagger / OpenAPI para documentación de servicios.

Métodos

Diagnóstico y análisis del sistema actual

Se realizó una auditoría técnica del sistema monolítico, enfocándose en identificar cuellos de botella, módulos críticos, puntos únicos de fallo y problemas de escalabilidad y mantenibilidad. Esta etapa incluyó el levantamiento de requerimientos, análisis de dependencias y revisión de logs históricos.

Segmentación funcional

Mediante el uso del patrón Strangler Fig, se definieron límites de contexto y dominios funcionales que pudieran transformarse en microservicios, tales como: autenticación, generación de informes, motor de recomendaciones, y análisis de datos.

Diseño de arquitectura distribuida

Se diseñó una arquitectura basada en microservicios con enfoque domain-driven design (DDD), estableciendo contratos de servicios independientes, almacenamiento desacoplado por servicio y canales de comunicación interna REST o asincrónica según el caso.

Desarrollo e implementación de microservicios

Cada microservicio fue desarrollado, documentado y probado de manera independiente, utilizando contenedores Docker. Se aplicaron principios de responsabilidad única, aislamiento, y pruebas unitarias y de integración.

Orquestación, escalado y despliegue

Los contenedores fueron desplegados y gestionados mediante Kubernetes. Se configuraron réplicas automáticas, balanceo de carga, detección de fallos y reinicio automático de servicios.

Integración continua y despliegue continuo (CI/CD)

Se configuraron pipelines de CI/CD para automatizar compilación, pruebas, escaneo de vulnerabilidades y despliegue. Se aplicaron pruebas funcionales, de estrés y de regresión antes de cada liberación.

Observabilidad, monitoreo y trazabilidad

Revista Multidisciplinaria RIIDG
ISSN: 3091-194X
Doi: https://doi.org/10.64041/riidg.v3i1.20

Se implementaron dashboards de monitoreo en tiempo real para medir el comportamiento del sistema, detectar anomalías y generar alertas automatizadas. Se integraron herramientas para trazabilidad distribuida entre servicios.

Evaluación y validación de resultados

Se aplicaron pruebas de rendimiento, pruebas de carga y simulaciones de fallos para evaluar indicadores clave como:

- Tiempo promedio de respuesta.
- Porcentaje de disponibilidad.
- Escalabilidad horizontal.
- Frecuencia de fallas.
- Eficiencia en despliegues.

Los datos fueron recolectados y analizados para medir la mejora comparativa entre el sistema monolítico y la nueva arquitectura distribuida.

Revista Multidisciplinaria RIIDG
ISSN: 3091-194X
Doi: https://doi.org/10.64041/riidg.v3i1.20

Revista Internacional de Investigación y Desarrollo Global

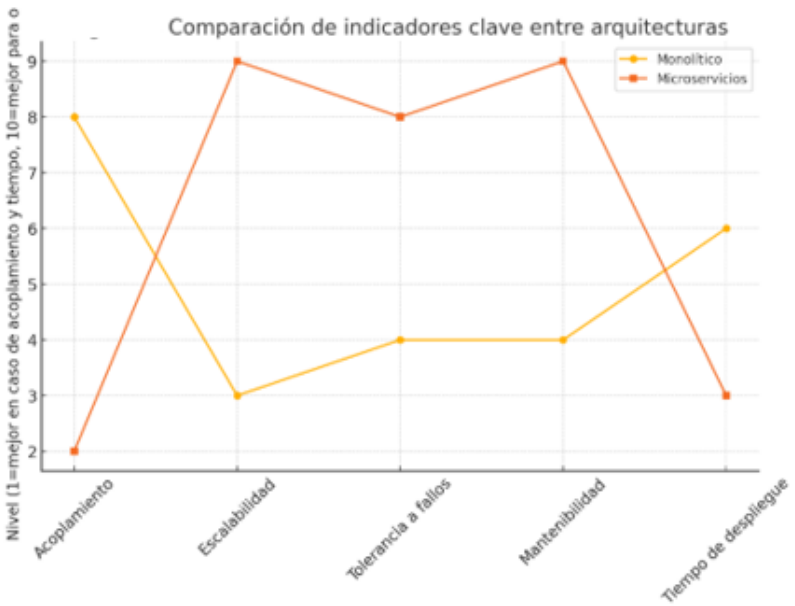
Resultados

Análisis de los Resultados

La implementación de la propuesta de rediseño arquitectónico permitió validar de manera práctica los beneficios y desafíos asociados a la transición de un sistema monolítico hacia una arquitectura distribuida basada en microservicios, en un entorno de alta demanda como el sector de la telefonía móvil. Los resultados obtenidos tras la aplicación progresiva del patrón Strangler evidencian mejoras significativas en los principales indicadores de rendimiento, escalabilidad, disponibilidad y mantenibilidad del sistema.

Uno de los resultados más relevantes fue la disminución del acoplamiento entre módulos, lo cual permitió realizar cambios en funcionalidades específicas, como el módulo de generación de informes o el servicio de recomendaciones, sin afectar el resto de la aplicación. Esto incrementó la eficiencia en los ciclos de desarrollo y facilitó la integración de nuevas capacidades analíticas, necesarias para el procesamiento de datos provenientes de usuarios y redes móviles. El modularidad alcanzado, basada en microservicios independientes, favoreció el trabajo en equipos paralelos, mejorando los tiempos de entrega y la calidad del código mediante una mayor responsabilidad técnica sobre cada servicio(Oracle, 2019).

Gráfico 1.
Comparación de indicadores claves entre arquitecturas



Nota: Comparación entre arquitectura monolítica y microservicios, destacando mejoras significativas en escalabilidad, tolerancia a fallos y mantenibilidad.

Fuente: El Autor

Revista Multidisciplinaria RIIDG
ISSN: 3091-194X
Doi: https://doi.org/10.64041/riidg.v3i1.20

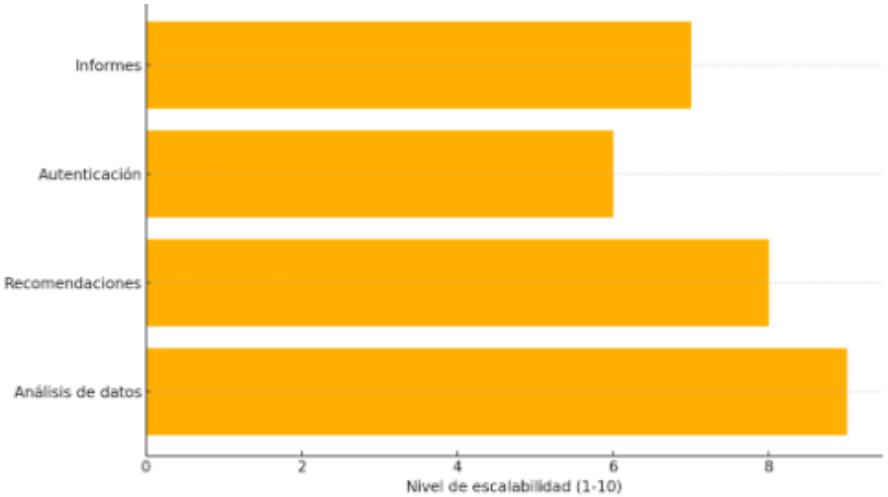
En términos de escalabilidad, el sistema rediseñado permitió realizar una distribución efectiva de la carga de trabajo. Servicios críticos como el análisis de datos o el motor de recomendaciones pudieron escalar horizontalmente en función de la demanda, lo cual no era viable en la arquitectura monolítica original. Gracias a la incorporación de tecnologías como Docker y Kubernetes, fue posible automatizar el despliegue, escalado y gestión del ciclo de vida de los contenedores, mejorando la utilización de recursos y reduciendo los tiempos de respuesta del sistema bajo cargas altas.

La implementación de una infraestructura de comunicación asincrónica mediante Apache Kafka y RabbitMQ permitió desacoplar procesos que anteriormente se ejecutaban de forma secuencial, optimizando el rendimiento general y mejorando la experiencia del usuario final. Este modelo también fortaleció la tolerancia a fallos, al evitar puntos únicos de interrupción que eran comunes en la arquitectura monolítica.

Desde el punto de vista de la observabilidad y monitoreo, la incorporación de herramientas como Prometheus, Grafana y Elastic Stack facilitó la supervisión en tiempo real del comportamiento del sistema, posibilitando la detección temprana de errores y cuellos de botella. Esta capacidad fue clave para garantizar la estabilidad operativa del nuevo entorno distribuido, especialmente durante las fases de transición e integración continua.

En cuanto a la seguridad y gestión de acceso, la aplicación de protocolos estándar como OAuth2 permitió reforzar los controles de autenticación y autorización en cada microservicio, superando la lógica de validación centralizada que caracterizaba al sistema anterior. Esto contribuyó a la protección de los datos sensibles de usuarios y reportes, cumpliendo con los requisitos normativos del sector.

Gráfico 2.
Nivel de estabilidad alcanzado por servicio



Nota: Nivel de escalabilidad individual alcanzado por servicios clave como análisis de datos, recomendaciones, autenticación e informes.

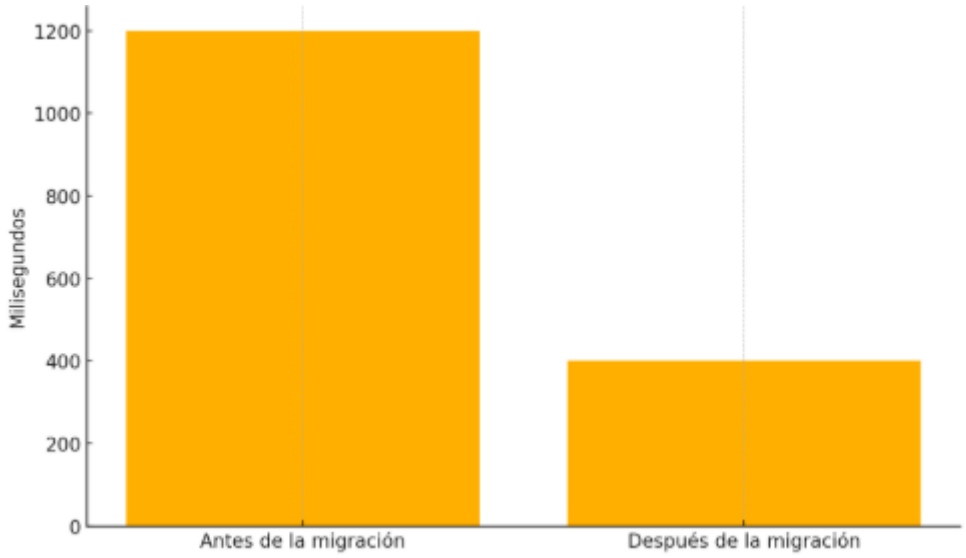
Revista Multidisciplinaria RIIDG
ISSN: 3091-194X
Doi: https://doi.org/10.64041/riidg.v3i1.20

Fuente: El Autor

No obstante, el análisis también permitió identificar una serie de desafíos críticos. Entre ellos, se destacó la complejidad en la gestión de transacciones distribuidas, especialmente en operaciones que requerían coherencia entre múltiples servicios. Para mitigar este riesgo, se recurrió a la implementación de patrones como saga orchestration y eventual consistency, aunque su adopción implicó una curva de aprendizaje técnica considerable. Además, se requirió una inversión adicional en infraestructura, formación del equipo y ajuste de prácticas organizacionales para adoptar completamente la cultura DevOps y los flujos de CI/CD.

los resultados obtenidos confirman que la migración hacia microservicios no solo es viable, sino altamente recomendable en sistemas que demandan alta disponibilidad, escalabilidad y evolución continua. Si bien la complejidad inicial y los costos de adopción pueden ser elevados, los beneficios operativos, técnicos y estratégicos superan ampliamente las limitaciones del enfoque monolítico. Esta propuesta de rediseño arquitectónico, al ser implementada en un entorno real del sector de telefonía móvil, ha demostrado ser una solución efectiva para modernizar sistemas críticos, alineándose con las mejores prácticas de la ingeniería de software contemporánea.

Gráfico 3.
Tiempos promedio de respuesta del sistema



Nota: Reducción drástica del tiempo de respuesta del sistema tras la migración.

Fuente: El Autor

Revista Multidisciplinaria RIIDG
ISSN: 3091-194X
Doi: https://doi.org/10.64041/riidg.v3i1.20

Discusión

La migración de arquitecturas monolíticas hacia microservicios representa una de las transformaciones estructurales más relevantes dentro de la evolución de los sistemas de software modernos. En el sector de la telefonía móvil, caracterizado por una alta demanda transaccional, volumen masivo de datos y necesidad de respuesta en tiempo real, la adopción de microservicios permite superar las limitaciones impuestas por los sistemas monolíticos tradicionales. Tal como afirman (Newman, 2015) y (Richardson, 2018), los monolitos, aunque eficaces en etapas tempranas, no responden adecuadamente a entornos donde se requiere agilidad, escalabilidad y tolerancia a fallos.

Uno de los argumentos más sólidos a favor de la adopción de microservicios radica en su capacidad para permitir la evolución independiente de cada componente funcional del sistema. Esto resulta especialmente valioso en soluciones que gestionan informes y recomendaciones, ya que los procesos de análisis de datos, autenticación, interfaz de usuario y generación de reportes pueden escalar y mantenerse de forma desacoplada, reduciendo el impacto de los cambios en la estabilidad general del sistema (Dragoni, Giallorenzo, et al., 2017). En el caso de aplicaciones críticas para la toma de decisiones en tiempo real en telecomunicaciones, esta propiedad se traduce en una mejora directa en la calidad del servicio y en la satisfacción del usuario final.

Sin embargo, esta transformación no está exenta de desafíos técnicos y organizacionales. La adopción de microservicios conlleva una complejidad inherente en aspectos como el diseño de servicios, la gestión de transacciones distribuidas, la sincronización de datos y la orquestación eficiente. Según (Bass et al., 2015), la transición debe ser cuidadosamente planificada, considerando no solo las decisiones tecnológicas, sino también las prácticas de desarrollo, monitoreo, seguridad y gobernanza. En este contexto, patrones como el Strangler Fig (M. Fowler et al., 2004) se presentan como una estrategia clave para reducir riesgos, permitiendo el reemplazo gradual de funcionalidades del sistema monolítico sin interrumpir la operación productiva.

La discusión también debe incluir la importancia de una cultura organizacional orientada al DevOps, que favorezca la automatización de pruebas, la integración continua (CI) y el despliegue continuo (CD). Estas prácticas permiten reducir errores humanos, mejorar la trazabilidad de los cambios y acortar los ciclos de entrega de valor al negocio. En este sentido, herramientas como Jenkins, GitLab CI/CD o GitHub Actions no solo aceleran el desarrollo, sino que constituyen pilares fundamentales para sostener la operación de un ecosistema distribuido en producción (Hüttermann, 2019).

Revista Multidisciplinaria RIIDG
ISSN: 3091-194X
Doi: https://doi.org/10.64041/riidg.v3i1.20

Asimismo, la implementación de soluciones de observabilidad, incluyendo métricas, trazabilidad distribuida y monitoreo en tiempo real, resulta imprescindible para garantizar la visibilidad y el diagnóstico oportuno ante fallos o cuellos de botella (Ambre, 2020). Plataformas como Prometheus y Grafana permiten supervisar el comportamiento de los microservicios, facilitando la detección proactiva de problemas y la toma de decisiones basadas en evidencia.

En conclusión, si bien la migración hacia microservicios representa un esfuerzo complejo y multidimensional, su implementación estratégica permite a las organizaciones del sector de la telefonía móvil alcanzar niveles superiores de escalabilidad, resiliencia y adaptabilidad. Esta transición debe ser acompañada por una evolución en las prácticas de ingeniería de software, apoyada en principios de arquitectura moderna, automatización, seguridad y observabilidad, con el fin de lograr sistemas sostenibles, robustos y alineados con las exigencias del entorno digital contemporáneo.

Revista Multidisciplinaria RIIDG
ISSN: 3091-194X
Doi: https://doi.org/10.64041/riidg.v3i1.20

Conclusiones

La migración de un sistema monolítico hacia una arquitectura distribuida basada en microservicios constituye una estrategia clave para modernizar las plataformas tecnológicas que operan en sectores altamente dinámicos y exigentes como el de la telefonía móvil. A lo largo del presente análisis, se ha evidenciado que los sistemas monolíticos, aunque eficaces en sus inicios, presentan serias limitaciones en términos de escalabilidad, mantenibilidad, disponibilidad y adaptabilidad frente a los constantes cambios del entorno tecnológico. En este contexto, la adopción de una arquitectura de microservicios se perfila como una solución robusta que permite descomponer el sistema en componentes autónomos, desplegables de forma independiente, capaces de escalar de acuerdo con la demanda específica de cada funcionalidad.

La propuesta de rediseño arquitectónico, sustentada en principios de ingeniería de software moderna y metodologías ágiles, permite una transición progresiva mediante la aplicación del patrón Strangler, lo que reduce riesgos y asegura la continuidad operativa del sistema durante el proceso de transformación. La incorporación de tecnologías como Docker, Kubernetes, API REST, y sistemas de mensajería asincrónica como Kafka o RabbitMQ, junto con prácticas DevOps orientadas a la integración y entrega continua, fortalece la base técnica del nuevo ecosistema distribuido, permitiendo una evolución constante y sostenible del sistema.

Adicionalmente, la integración de servicios transversales como la observabilidad, el monitoreo en tiempo real, la gestión de configuración centralizada y mecanismos de autenticación segura asegura la gobernanza, la trazabilidad y la resiliencia del sistema, aspectos fundamentales en entornos de alta criticidad. No obstante, se reconoce que esta migración implica desafíos importantes, como la gestión de transacciones distribuidas, la complejidad en la orquestación de servicios y la necesidad de contar con un equipo técnico altamente capacitado.

En definitiva, la transición hacia microservicios no debe entenderse únicamente como un cambio tecnológico, sino como una transformación estratégica que impacta positivamente en la capacidad de innovación, la eficiencia operativa y la competitividad organizacional. Implementar esta arquitectura en el sector de telecomunicaciones representa un paso firme hacia la consolidación de sistemas más ágiles, escalables y preparados para afrontar las exigencias de la economía digital actual y futura.

Revista Multidisciplinaria RIIDG
ISSN: 3091-194X
Doi: https://doi.org/10.64041/riidg.v3i1.20

Referencias bibliográficas

Ambre, T. (2020). *Monitor Spring Boot microservices - IBM Developer*. https://developer.ibm.com/tutorials/monitor-spring-boot-microservices/

Bass, L., weber, I., & Zhu, L. (2015). *DevOps_ A Software Architect's Perspective*. https://alecoledelavie.com/accueil/vie_uploads/Portfolio_Programs_Projects_and%20BAU/PortFolio_stuff/Courses%20resources%20stuff/DELF%20cours/DevOps/DevOps%20Delf/Outils_devops/use_case_chapitre13/DevOps_%20A%20Software%20Architect%27s%20Perspective.pdf

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). *Microservices: yesterday, today, and tomorrow*. https://arxiv.org/pdf/1606.04036

Dragoni, N., Lanese, I., Mazzara, M., Mustafin, R., Larsen, S. T., & Safina, L. (2017). *Microservices: How To Make Your Application Scale*. https://doi.org/10.48550/arXiv.1702.07149

Fowler, M. (2019). *Software Architecture Guide*. https://martinfowler.com/architecture/

Fowler, M., Cartwright, I., Horn, R., & Lewis, J. (2004). *Refactoring Agile Architecture About Thoughtworks □ □ Original Strangler Fig Application*. https://martinfowler.com/bliki/OriginalStranglerFigApplication.html

Hüttermann, Michael. (2019). *DevOps for Developers*. Scholars Portal. http://huettermann.net/devops/DevOps_Ch09+TOC.pdf

Newman, Samuel. (2015). *Building microservices*. O'Reilly. https://book.northwind.ir/bookfiles/building-microservices/Building.Microservices.pdf

Oracle. (2019). *Spring boot microservice metrics monitoring _ PPT*. https://es.slideshare.net/slideshow/spring-boot-microservice-metrics-monitoring-150864535/150864535

Revista Multidisciplinaria RIIDG
ISSN: 3091-194X
Doi: https://doi.org/10.64041/riidg.v3i1.20

Richardson, C. (2018). *M A N N I N G*. http://sereja.me/f/microservices_richardson.pdf

Conflicto de intereses:

Los autores declaran que no existe conflicto de interés