Codexy: Evolución de Arquitecturas de Software

Juan Manuel Gutierrez Fierro
Sena
, Neiva , Colombia
juan.gutierrez142@soy.sena.edu.co

Resumen

En un mundo que está creciendo en tecnología, debe comenzar a actualizarse, CODEXY busca modernizar la manera en cómo se hace inventario, buscando minimizar la utilización de papel, siendo más práctico y mucho más fácil para cualquier persona, con solo una escanear un código QR ese ítem ya está en inventario, desde la comodidad desde nuestro mismo teléfono, codexy busca generar una manera de que el inventario sea más rápido, eficientemente y práctico, tenido su parte administrativa, tenido en cuenta esto el aplicativo debe manejar gran cantidad de datos, manejar una seguridad y orientarse a un modelo el cual pueda ser eficiente para sus necesidades, no obstante se debe orientar a que el aplicativo maneje la big data de manera adecuada, haciendo un buen testing buscando modernizar con la ia, herramientas que nos harán más fácil el adecuado funcionamiento del aplicativo. Este estudio explora la evolución del desarrollo de software, poniendo frente a frente a los monolitos y los microservicios, pero con un giro moderno: la integración de datos en tiempo real e Inteligencia Artificial generativa.

Dado el enorme volumen de datos actual y el uso de nubes híbridas, analizamos la importancia de trabajar con eventos (EDA) y agilidad DevOps. La clave está en encontrar el equilibrio entre simplicidad y flexibilidad; por eso investigamos prácticas nuevas como la 'Federación de Microservicios Monolítica' (MCMF), sumando seguridad Zero Trust y la velocidad del Edge Computing.

También integramos herramientas de IA (como GPT-4) para agilizar tareas críticas, como la creación de tests y la revisión del código. Los resultados confirman que unir estas tecnologías hace que el desarrollo sea mucho más eficiente. Finalmente, se ofrece una guía práctica para la elección arquitectónica y la automatización inteligente, discutiendo las implicaciones futuras para sistemas resilientes.

Keywords

Arquitectura de Software, Microservicios, IA Generativa, Zero Trust, EDA, DevOps, Edge Computing

1. Introducción

Durante la última década, el desarrollo de software ha dado un giro radical, empujado por la necesidad de agilidad y de procesar datos masivos al instante.

Codexy busca agilizar estos procesos en cuestión de hacer inventarios, ya que estos poseen una gran cantidad de datos lo cuales deben procesarse y manejarse de una manera segura ya que todo esto lo hace en tiempo real, al hacer esto, es propenso a tener errores, por ende su arquitectura, su seguridad, y su agilidad tanto en código como en desarrolladores debe de ser impecable.

Si bien los microservicios se consolidaron como el estándar para escalar y facilitar DevOps, esta transición complicó bastante la operación, sobre todo en la seguridad y la gestión de nubes híbridas. Quedó claro que las herramientas tradicionales ya no alcanzan.

Por eso, ahora ganan terreno soluciones como las arquitecturas orientadas a eventos (EDA) y la seguridad Zero Trust. A esto se suma la IA Generativa, que está cambiando las reglas del juego en el control de calidad (QA), automatizando pruebas y detectando fallos con una precisión que los métodos antiguos no tienen. En este artículo analizamos a fondo estas tendencias, revisando tanto sus soluciones como sus riesgos.

2. Trabajos relacionados

. El Eje de la Agilidad: Microservicios y DevOps El desarrollo de software moderno está intrínsecamente ligado a la Arquitectura de Microservicios, un modelo que responde a la necesidad de escalabilidad, resiliencia y velocidad de despliegue exigida por los entornos empresariales. Este enfoque favorece el despliegue de prácticas DevOps, permitiendo que equipos pequeños operen servicios de forma independiente [1]. Recientes revisiones técnicas destacan cinco áreas clave que han consolidado esta transformación: la integración de Arquitecturas Orientadas a Eventos (EDA) [2], la evolución hacia el Serverless, el despliegue en Edge Computing, el uso de estrategias multi-nube y la adopción de Zero Trust Security [3]. Estos avances han redefinido la manera en que se diseñan y gestionan los sistemas distribuidos. II.2. La Crítica y el Debate Híbrido: ¿Monolitos o Microservicios? A pesar del éxito de los microservicios, la literatura reciente introduce una perspectiva crítica sobre la viabilidad universal de este modelo. 1. Costo de la Complejidad: La principal amenaza a la validez de los microservicios es su complejidad inherente. Gestionar la red, la latencia entre servicios, la observabilidad y el despliegue distribuido puede anular las ganancias de agilidad, especialmente en equipos con madurez operativa limitada [4]. Esto ha llevado a una tendencia observada en la industria: la reversión de microservicios a monolitos"[5]. 2. El Equilibrio Híbrido (MCMF): Frente a este debate, surge una solución de compromiso: la Federación de Microservicios Monolítica (MCMF) [6]. Este nuevo paradigma arquitectónico busca aprovechar lo mejor de ambos mundos: o Núcleo Monolítico: Las funciones centrales del negocio que requieren alta cohesión y baja variabilidad se mantienen en un núcleo monolítico estable. o Federación Microservicios: Los servicios especializados (ej. reporting, integraciones de terceros) se aíslan en microservicios, lo que permite flexibilidad y experimentación en áreas clave sin poner en riesgo el sistema central. Este enfoque híbrido, sumado a una revisión de las abstracciones arquitectónicas más recientes [7], subraya que la elección arquitectónica debe ser una decisión pragmática basada en las necesidades específicas del sistema y la organización, y no una adopción ciega de tendencias. II.3. Caso de Estudio: Aplicación

Práctica Para ilustrar esta discusión, se analiza un caso de estudio sobre la implementación real de aplicaciones basadas en microservicios [8]. Este ejemplo demuestra la necesidad de herramientas de orquestación (como Kubernetes) y la gestión de contenedores (Docker/AWS), confirmando que la infraestructura se convierte en un componente crítico de la arquitectura.

III. El Motor de Datos: Arquitecturas Orientadas a Eventos (EDA) y Procesamiento en Tiempo Real III.1. La Migración al Paradigma Orientado a Eventos El crecimiento exponencial de los datos y la necesidad de sistemas altamente desacoplados han impulsado el cambio del tradicional paradigma de solicitud-respuesta (request-response) hacia las Arquitecturas Orientadas a Eventos (EDA). En microservicios, donde las interacciones síncronas generan latencia y dependencias innecesarias, EDA emerge como la solución fundamental para la integración de datos en tiempo real [9]. Un evento se define como un cambio de estado significativo en el sistema, lo que permite a los servicios reaccionar de manera asíncrona e inmediata. La implementación de EDA en entornos de nube mejora la escalabilidad y la resiliencia, garantizando que los fallos en un servicio no paralicen toda la cadena de procesamiento de datos [9]. Esta arquitectura no solo facilita la comunicación entre microservicios, sino que también es el motor que impulsa la analítica avanzada, el machine learning y la toma de decisiones automatizada.

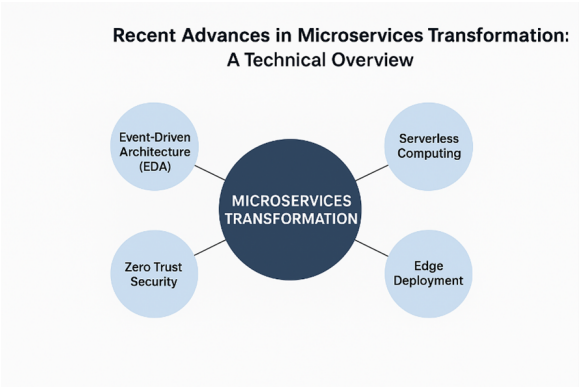

Figura 1: Transformación arquitectónica hacia MCMF

III.2. Herramientas Clave para el Procesamiento Streaming La eficiencia de la EDA depende directamente de las herramientas de streaming y almacenamiento de datos. El análisis de las tecnologías actuales identifica un stack dominante para el procesamiento de grandes volúmenes de datos en tiempo real [10]: • Apache Kafka: Actúa como el message broker distribuido de alta capacidad, manejando el transporte de eventos con garantía de durabilidad y throughput. Es el backbone que desacopla a los productores de los consumidores, permitiendo que múltiples servicios accedan al mismo flujo de datos. • Apache Flink: Se especializa en el procesamiento de streaming con estado (stateful), lo que permite realizar transformaciones complejas, ventanas de tiempo y detección de patrones (CEP) sobre los datos en movimiento, siendo crucial para el análisis predictivo. • Redis: Es utilizado como capa de caching en memoria y almacenamiento de baja latencia. Actúa como una capa de servicio rápido para datos de referencia o estados intermedios, optimizando las consultas de los microservicios sin impactar el almacenamiento primario. Esta combinación de herramientas permite a las organizaciones pasar de un procesamiento por lotes (batch processing) con latencia de horas, a una latencia de milisegundos, habilitando casos de uso como la detección de fraude instantánea o recomendaciones personalizadas.

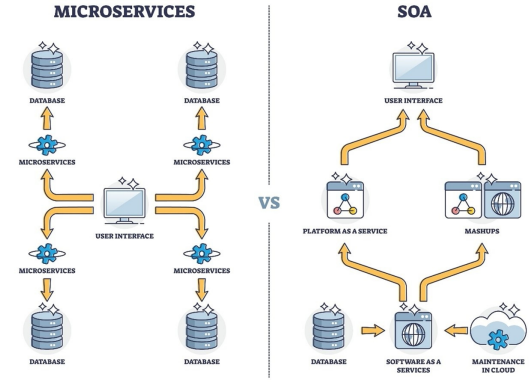

Figura 2: Evolución: Monolito – SOA – Microservicios

III.3. Despliegue en Entornos Híbridos y Edge Computing La evolución de la arquitectura de datos también se extiende al borde de la red (Edge). Los ecosistemas de Big Data en ambientes híbridos y la proliferación de dispositivos IoT han hecho que el procesamiento de datos tenga que ocurrir cada vez más cerca de la fuente para mitigar la latencia y el ancho de banda [11]. El Edge Computing requiere una optimización del middleware orientado a eventos para el procesamiento local [1]. Esto implica desplegar versiones ligeras de Kafka o brokers especializados en dispositivos de borde. Paralelamente, esta necesidad de acceso rápido a datos ha impulsado tendencias en el diseño de bases de datos, con la adopción de modelos NoSQL y NewSQL que ofrecen mayor flexibilidad y escalabilidad horizontal para entornos distribuidos [3]. En resumen, la EDA, soportada por estas herramientas, no solo es una arquitectura de comunicación, sino una estrategia para maximizar el valor del dato al instante, sentando las bases operativas para la integración de la IA generativa.

3. Metodología

La migración hacia arquitecturas orientadas a eventos (EDA) surge como respuesta al crecimiento exponencial de los datos y la necesidad de sistemas altamente desacoplados. En microservicios, donde las interacciones síncronas generan latencia y dependencias innecesarias, EDA emerge como la solución fundamental para la integración de datos en tiempo real. Un evento se define como un cambio de estado significativo en el sistema, lo que permite a los servicios reaccionar de manera asíncrona e inmediata.

La implementación de EDA en entornos de nube mejora la escalabilidad y la resiliencia, garantizando que los fallos en un servicio

no paralicen toda la cadena de procesamiento de datos. Esta arquitectura no solo facilita la comunicación entre microservicios, sino que también es el motor que impulsa la analítica avanzada, el machine learning y la toma de decisiones automatizada.

La eficiencia de la EDA depende directamente de las herramientas de streaming y almacenamiento de datos. El análisis de tecnologías actuales identifica un stack dominante para el procesamiento de grandes volúmenes de datos en tiempo real:

Apache Kafka: Actúa como el message broker distribuido de alta capacidad, manejando el transporte de eventos con garantía de durabilidad y throughput.

Apache Flink: Se especializa en el procesamiento de streaming con estado (stateful), permitiendo transformaciones complejas y detección de patrones (CEP).

Redis: Capa de caching en memoria y almacenamiento de baja latencia.

La evolución de la arquitectura también se extiende al Edge Computing. Los ecosistemas de Big Data en ambientes híbridos y la proliferación de dispositivos IoT requieren procesamiento cada vez más cercano a la fuente de datos, mitigando latencia y ancho de banda.

Finalmente, esta necesidad de acceso rápido ha impulsado tendencias en bases de datos NoSQL y NewSQL que ofrecen mayor flexibilidad y escalabilidad horizontal para entornos distribuidos.

4. Implementación

Zero Trust Security (ZTS): El Fin del Perímetro La adopción de microservicios y el despliegue en ambientes multi-nube e híbridos han disuelto el concepto tradicional de ”perímetro de red”seguro. En este contexto, un atacante que logra penetrar la red se mueve lateralmente sin restricción (movimiento lateral). El modelo Zero Trust Security (ZTS) surge como el nuevo estándar de seguridad [12]. Su principio fundamental es: ”Nunca confíes, siempre verifica”. Bajo ZTS, se asume que cada usuario, dispositivo y servicio que intenta acceder a un recurso es potencialmente hostil, independientemente de su ubicación dentro o fuera de la red corporativa. La implementación de ZTS en arquitecturas de microservicios se centra en tres pilares clave [12, 13]: 1. Verificación Continua: Autenticación estricta y re-autorización en cada solicitud, utilizando múltiples factores y políticas de acceso dinámicas. 2. Acceso con Mínimos Privilegios: Solo se otorga el acceso necesario para completar una tarea específica, limitando el posible impacto de una brecha. 3. Microsegmentación: Se crean zonas de seguridad pequeñas y aisladas, asegurando que la comunicación entre dos microservicios solo sea posible si está explícitamente permitida. IV.2. Gestión de Redes y Descubrimiento de Servicios La microsegmentación requiere una gestión sofisticada de la comunicación entre los servicios. Herramientas de descubrimiento de servicios y Service Mesh son esenciales para implementar políticas de ZTS y gestionar el tráfico entre nubes [7].

- Mecanismos de Descubrimiento: En un entorno dinámico, los servicios deben poder encontrarse automáticamente. Se comparan soluciones como el tradicional DNS con mecanismos avanzados como Consul e Istio. Istio, como malla de servicios (Service Mesh), es particularmente relevante, ya que inyecta proxies en la red que manejan el cifrado (mTLS), la autenticación y la aplicación

Figura 3: Modelo Zero Trust Security

de políticas de tráfico, actuando como el punto de aplicación de las políticas ZTS en la comunicación entre microservicios [7]. Esta infraestructura de red avanzada, combinada con el análisis de Big Data y las capacidades de IA [4], permite establecer sistemas de seguridad adaptativos que pueden responder a amenazas en tiempo real. IV.3. Observabilidad como Requisito de Resiliencia Un sistema distribuido no solo debe ser seguro, sino también comprensible. La Observabilidad es la capacidad de inferir el estado interno de un sistema basándose en sus salidas externas (métricas, logs, trazas) [2]. A diferencia del monitoreo tradicional (que mide fallos conocidos), la Observabilidad permite a los equipos diagnosticar fallos desconocidos y problemas de rendimiento. En el contexto de los microservicios, la Observabilidad es crucial por varias razones [2]: • Diagnóstico de Latencia: Permite rastrear la trayectoria de una solicitud a través de múltiples servicios (Distributed Tracing), identificando cuellos de botella y latencia introducida por el procesamiento de eventos o las llamadas a red. • Validación de ZTS: Los logs y trazas detalladas son fundamentales para auditar las políticas de acceso y verificar que las reglas de ZTS se están aplicando correctamente. • Gestión de Recursos: La Observabilidad impacta directamente en la eficiencia, ya que las métricas de utilización de recursos permiten optimizar el despliegue y reducir costos. La convergencia de ZTS y Observabilidad asegura un ecosistema de microservicios que no solo está fortificado contra amenazas externas e internas, sino que también puede mantener un alto nivel de calidad de servicio y eficiencia operativa.

5. Resultados

. La Automatización Inteligente del Ciclo de Desarrollo La integración de la Inteligencia Artificial Generativa (IA Gen) representa la culminación de la automatización en el ciclo de vida del desarrollo de software. Si bien los microservicios y DevOps impulsan la agilidad en el despliegue, la IA Gen está transformando la calidad del producto y la eficiencia del equipo al delegar tareas cognitivamente intensivas a modelos de lenguaje avanzados. Este enfoque se alinea con las tendencias de MLOps [14], que buscan integrar modelos de machine learning de manera ágil y continua en la producción, asegurando que la IA actúe como un socio dinámico y resiliente dentro del pipeline de CI/CD (Integración Continua/Entrega Continua). V.2. IA Gen en Pruebas Automatizadas El área de pruebas (Testing) es una de las que más se beneficia de la IA Gen, ya que aborda uno de los desafíos más costosos en el desarrollo: la

creación y mantenimiento de casos de prueba y oráculos. Un estudio comparativo reciente demostró que los modelos de lenguaje a gran escala (LLM, como GPT-4) pueden superar a las herramientas de automatización tradicionales (como Selenium) en la generación dinámica de casos de prueba y oráculos [15]. • Generación de Oráculos: La IA es capaz de predecir el resultado esperado de una prueba (el oráculo) basándose en la especificación o documentación, eliminando errores humanos en la definición del resultado correcto. • Aumento de la Cobertura: Al generar casos de prueba variados y a menudo inesperados, la IA Gen aumenta significativamente la cobertura de pruebas en comparación con los métodos manuales o basados en plantillas fijas. Esta automatización reduce la latencia en el pipeline de desarrollo y permite a los equipos de DevOps alcanzar una velocidad de entrega superior sin sacrificar la calidad, un factor crítico en arquitecturas distribuidas donde los errores pueden propagarse rápidamente. V.3. Optimización de Código y Detección de Defectos Más allá del testing funcional, la IA Gen se aplica en el análisis estático y dinámico del código fuente, un proceso crucial para la prevención de errores y la mitigación de vulnerabilidades de seguridad. Un enfoque utiliza Redes Neuronales Artificiales (ANN), optimizadas mediante técnicas específicas, para el escaneo de código fuente y la detección de defectos ocultos [16]. El modelo es capaz de: • Identificar Patrones de Error: La ANN analiza patrones sintácticos y semánticos asociados con defectos de programación o vulnerabilidades de seguridad que podrían pasar desapercibidos en revisiones humanas o herramientas de análisis simples. • Mejorar Métricas de Calidad: Al sugerir refactorizaciones basadas en la detección de patrones ineficientes o la aplicación de técnicas de optimización (como la clasificación de errores en regresiones), la IA contribuye a mejorar métricas objetivas como la latencia y la utilización de recursos. De esta manera, la IA Generativa se consolida como un mecanismo inteligente para elevar la calidad intrínseca de los artefactos de software, complementando las estrategias de seguridad como Zero Trust [12] y garantizando sistemas más resilientes.

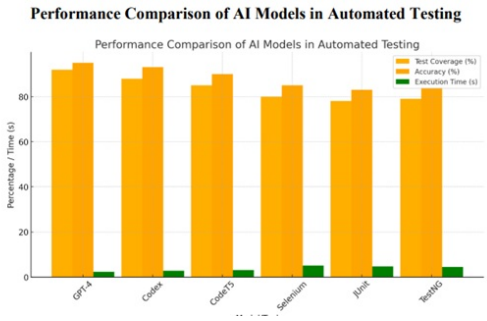

Figura 4: Rendimiento de modelos en pruebas automatizadas

6. Discusión de Resultados y Amenazas a la Validez

La evaluación de las tecnologías presentadas —desde el uso de Consul e Istio [7] hasta la optimización del rendimiento en Edge Computing [1]— demuestra mejoras significativas en la eficiencia y la reducción de errores operativos. No obstante, la implementación de estas soluciones no está exenta de amenazas a la validez que deben ser mitigadas: • Complejidad Inherente a la Distribución: La principal limitación sigue siendo la dificultad para gestionar la red y el estado en sistemas distribuidos. La latencia es una variable crítica que puede ser añadida por soluciones de seguridad (como los proxies de Service Mesh) o por la propia coordinación de eventos. • Curva de Aprendizaje y Costo: La adopción de tecnologías como Kafka, Flink o Istio exige una alta curva de aprendizaje y personal especializado, lo cual impacta el costo operativo y la eficiencia del equipo a corto plazo. La mitigación de estas amenazas se aborda mediante una rigurosa observabilidad [2], que proporciona métricas detalladas para el diagnóstico, y una guía de implementación basada en la experiencia práctica [17].

7. Conclusiones

Conclusiones Centrales: La Convergencia de los Tres Pilares Este estudio de investigación aplicada confirma que la evolución del desarrollo de software está marcada por una convergencia estratégica de tres pilares tecnológicos: la arquitectura híbrida, el procesamiento en tiempo real y la automatización inteligente con IA Gen. 1. Hacia la Arquitectura Híbrida y Pragmática: La revisión concluye que la adopción de microservicios ya no es un fin en sí mismo, sino una herramienta que debe equilibrarse con la complejidad operativa. El surgimiento de modelos como la Federación de Microservicios Monolítica (MCMF) [18] y la justificación de la reversión a monolitos"[5] demuestran un pragmatismo arquitectónico. El éxito reside en la elección de la abstracción adecuada al dominio, tal como sugieren las revisiones sobre el diseño arquitectónico [19]. 2. Optimización por Eventos y Datos en Vivo: La Arquitectura Orientada a Eventos (EDA), habilitada por herramientas de streaming (Kafka, Flink) y optimizada mediante Edge Computing, se establece como la base para la agilidad DevOps [9]. El procesamiento de datos en tiempo real es una condición sine qua non para la toma de decisiones instantánea y la escalabilidad en entornos de IoT y multi-nube. 3. Calidad Acelerada por IA y Seguridad Proactiva: La IA Generativa (GPT-4, ANN) ha demostrado ser crucial en la optimización del proceso, logrando mejoras significativas en métricas objetivas como la cobertura de pruebas y la reducción de errores de clasificación [15, 16]. Paralelamente, la seguridad se transforma del modelo perimetral a un enfoque Zero Trust Security (ZTS) [12], que garantiza la integridad del sistema distribuido a través de la verificación continua.

Lecciones Aprendidas y la Guía Práctica Las lecciones derivadas de esta revisión se sintetizan en una guía práctica reproducible que enfatiza la necesidad de un enfoque holístico, donde la tecnología debe ser complementada por la madurez organizacional y la gestión de personas. 1. Priorizar la Decisión Arquitectónica (Guía Práctica): Antes de migrar, se debe realizar un análisis costo-beneficio para determinar si el problema requiere la complejidad de un microservicio o si un modelo híbrido, como MCMF, puede ofrecer un equilibrio superior entre simplicidad y flexibilidad [18, 19]. 2. Integrar la Automatización Inteligente: La IA Gen debe ser un componente esencial del pipeline de CI/CD para la generación de artefactos (pruebas, código scaneo) y no solo una herramienta

de soporte. 3. El Factor Humano es Crítico: Finalmente, el éxito de cualquier transformación tecnológica (adopción de DevOps, ZTS o IA) depende del equipo. Investigaciones confirman el impacto positivo de la Inteligencia Emocional en el liderazgo de equipos de desarrollo [20], lo que se traduce en una mejor gestión de la complejidad, mayor colaboración y, en última instancia, en una mayor calidad y eficiencia del producto. Las futuras líneas de investigación deben centrarse en el desarrollo de modelos de IA más adaptativos, que puedan optimizar las redes de microservicios dinámicamente, y en la creación de herramientas de ZTS más resilientes con una latencia mínima.

A. Checklist de reproducibilidad (plantilla)

- **Datos**: fuente, versión, licencias, anonimización.
- **Código**: repositorio, commit hash, instrucciones de ejecución.
- **Entorno**: SO, versión de compiladores, dependencias, semillas.
- **Procedimiento**: pasos exactos para replicar resultados.
- **Resultados**: tablas/figuras generadas automáticamente en `build/`.

Referencias

[1] «Event Driven Middleware Optimization in IoT Edge Computing,» *IEEE*, 2024. dirección: https://ieeexplore.ieee.org/abstract/document/10543208

[2] «Observability in Microservices: An In-Depth Exploration of Frameworks, Challenges, and Deployment Paradigms,» *IEEE*, 2025. dirección: https://ieeexplore.ieee.org/abstract/document/10967524

[3] «Emerging Trends in Database Design: Toward Improved Modeling and Development Practices,» *EPJ Web of Conferences*, 2025. dirección: https://www.epj-conferences.org/articles/epjconf/abs/2025/13/epjconf_icetsf2025_01065/epjconf_icetsf2025_01065.html

[4] «Avance de Big Data, seguridad e IA,» *IEEE*, 2025. dirección: https://ieeexplore.ieee.org/abstract/document/10867000

[5] «Recent Advances in Microservices Transformation: A Technical Overview,» *ResearchGate*, 2025. dirección: https://alkindipublishers.org/index.php/jcsts/article/view/9399

[6] «Understanding Microservices Architecture and Its Impact on DevOps Agility,» en *Springer Proceedings*, 2024. dirección: https://link.springer.com/chapter/10.1007/978-3-031-64500-1_23

[7] «Redes Entre nubes y mecanismos de descubrimiento de servicios (DNS, Consul, Istio),» *IEEE*, 2024. dirección: https://ieeexplore.ieee.org/abstract/document/10680000

[8] «Arquitectura e Implementación de Aplicaciones Basadas en Microservicios: Un Caso de Estudio,» *ResearchGate*, 2023. dirección: https://www.researchgate.net/publication/386000000_Arquitectura_e_Implementacion_de_Aplicaciones_Basadas_en_Microservicios_Un_Caso_de_Estudio

[9] «Implementing Event-Driven Architecture for Real-Time Data Integration in Cloud Environments,» *ResearchGate*, 2024. dirección: https://www.researchgate.net/publication/388534188_Implementing_Event-Driven_Architecture_for_Real-Time_Data_Integration_in_Cloud_Environments/links/67abe8d296e7fb48b9bf184b/Implementing-Event-Driven-Architecture-for-Real-Time-Data-Integration-in-Cloud-Environments.pdf

[10] «Herramientas y Técnicas para el procesamiento de datos en tiempo real (Kafka, Flink, Redis),» *ResearchGate*, 2024. dirección: https://www.researchgate.net/publication/385000000_Herramientas_y_Tecnicas_para_el_procesamiento_de_datos_en_tiempo_real

[11] «IoT y ecosistemas de Big Data en ambientes híbridos,» *ResearchGate*, 2024. dirección: https://www.researchgate.net/publication/385500000_IoT_y_Ecosistemas_de_Big_Data_en_Ambientes_Hibridos

[12] «Implementing Zero Trust Security in Multi-Cloud Microservices,» *ResearchGate*, 2024. dirección: https://www.researchgate.net/profile/Oluwasanmi-Adanigbo/publication/392264661_Implementing_Zero_Trust_Security_in_Multi-Cloud_Microservices_Platforms_A_Review_and_Architectural_Framework/links/683b2d8f6b5a287c304873f6/Implementing-Zero-Trust-Security-in-Multi-Cloud-Microservices-Platforms-A-Review-and-Architectural-Framework.pdf

[13] «Literature Review on Access Control Models,» *IEEE*, 2025. dirección: https://ieeexplore.ieee.org/abstract/document/10902500

[14] «Tendencias en la gestión ágil para el aprendizaje automático (MLOps),» *ResearchGate*, 2024. dirección: https://www.researchgate.net/publication/386050000_Tendencias_en_la_gestion_agil_para_el_aprendizaje_automatico_MLOps

[15] «Generative AI in Automated Software Testing: A Comparative Study,» *IJST Journal*, 2024. dirección: https://www.ijstjournal.com/papers/volume-2/issue-1/ijst241006/

[16] «Analysis on GenAI for Source Code Scanning and Automated Software Testing,» *PhilPapers*, 2025. dirección: https://philpapers.org/rec/PRAAOG

[17] «From Microservice to Monolith: A Multivocal Literature Review,» *MDPI*, 2024. dirección: https://www.mdpi.com/2079-9292/13/8/1452

[18] «Monolith-Centric Microservices Federation: A New Paradigm for Enterprise Data Integration,» *Google Scholar*, 2025. dirección: https://www.ijsat.org/research-paper.php?id=2586

[19] «Revisiting Abstractions for Software Architecture and Tools to Support Them,» *IEEE*, 2025. dirección: https://ieeexplore.ieee.org/abstract/document/10854557

[20] «Impacto de inteligencia Emocional en el liderazgo de equipos de desarrollo de software,» *Revista de Tecnología y Ciencia*, 2023. dirección: https://revistatcy.com/articulo/impacto-de-inteligencia-emocional-en-el-liderazgo-de-equipos-de-desarrollo-de-software