Yanuard Stevin montialegre Bonilla



Soy un desarrollador full-stack con una profunda pasión por el mundo del desarrollo de software. Desde que comencé mi camino en la tecnología, me he dedicado a aprender y dominar diversas herramientas y lenguajes de programación, siempre con el objetivo de crear aplicaciones que no solo sean funcionales, sino también eficientes y escalables. Mi curiosidad y deseo constante de mejorar me han llevado a explorar una amplia variedad de tecnologías, tanto en el frontend como en el backend, lo que me ha permitido desarrollar una visión integral de los proyectos en los que trabajo.

Mi enfoque se basa en utilizar arquitecturas de software modernas y seguir las mejores prácticas del desarrollo, asegurándome de que cada solución que implemento sea robusta, segura y adaptable a las necesidades cambiantes del negocio. A lo largo de mi carrera, he trabajado con diferentes frameworks, bibliotecas y metodologías ágiles, lo que me ha brindado una gran flexibilidad y capacidad para adaptarme rápidamente a nuevos desafíos.

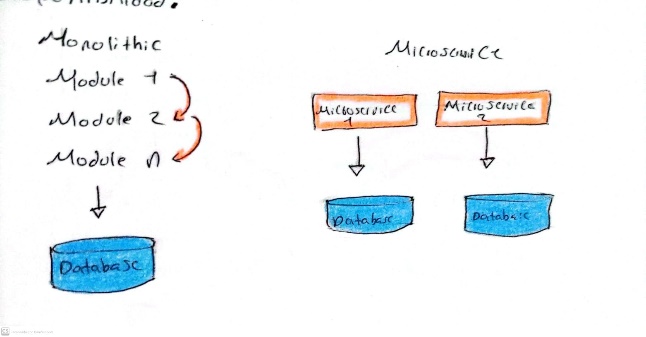
En definitiva, mi pasión por la tecnología y mi compromiso con el aprendizaje continuo me impulsan a dar lo mejor de mí en cada proyecto.

**Development of a Quality-Based Model for Software Architecture Optimization: A Case Study of Monolith and Microservice Architectures**

El artículo compara las arquitecturas monolíticas y de microservicios en términos de su impacto en la calidad del software. Analiza seis atributos de calidad: acoplamiento, capacidad de prueba, seguridad, complejidad, capacidad de implementación y disponibilidad. El artículo destaca un "punto de intersección" donde las arquitecturas monolíticas y de microservicios tienen métricas de calidad de software idénticas, lo cual ocurre solo cuando hay una característica. Además, explora las compensaciones entre las arquitecturas, donde los microservicios pueden mejorar la capacidad de implementación y la disponibilidad, pero afectan negativamente la capacidad de prueba, el acoplamiento y la complejidad. Se proporcionan recomendaciones para elegir la mejor arquitectura en función de la importancia de atributos de calidad específicos. El artículo enfatiza la importancia de la evaluación continua de la calidad durante todo el proceso de desarrollo de software. Finalmente, los autores concluyen que la mejor opción entre las arquitecturas monolíticas y de microservicios depende de los requisitos y el contexto específico del sistema.

Reflexión:

Este texto subraya la importancia de tomar decisiones arquitectónicas basadas en la calidad del software y en el contexto específico de cada sistema. La comparación entre arquitecturas monolíticas y de microservicios revela que no hay una solución perfecta, sino compensaciones en atributos clave como la capacidad de prueba, seguridad y disponibilidad. La reflexión central es que la elección de la mejor arquitectura debe adaptarse a las prioridades y necesidades del proyecto, y que la evaluación continua de la calidad es crucial para mantener un software eficiente y sostenible.



# 

# **Microservice Architecture is Agile Software Architecture**

Desde que el concepto de microservicios irrumpió en la industria del software en 2014, los profesionales han estado examinando esta nueva arquitectura. Este estilo arquitectónico surgió de patrones comunes observados en empresas como Amazon, Netflix y SoundCloud. Los microservicios fueron una evolución de las aplicaciones monolíticas hacia sistemas compuestos por servicios individuales que se comunican a través de APIs RESTful, favoreciendo el desarrollo ágil, la entrega continua y la cultura DevOps.

El Manifiesto Ágil de 2001 marcó un cambio en la forma de desarrollar software, enfocado en eliminar el riesgo de los grandes proyectos mediante iteraciones pequeñas y colaborativas. Con el tiempo, se popularizó la integración continua (CI) para reducir problemas de integración de código y, posteriormente, la entrega continua (CD) para acelerar los lanzamientos. A pesar de estos avances, las organizaciones se encontraron con un cuello de botella entre el desarrollo y las operaciones, lo que dio origen al movimiento DevOps, que fomenta la colaboración entre ambos equipos.

Finalmente, muchas de estas empresas, impulsadas por la necesidad de velocidad y escalabilidad, descubrieron que una arquitectura de microservicios era más adecuada para sus prácticas ágiles y DevOps, ya que permitía que los componentes pequeños y desacoplados fueran desarrollados, probados y escalados de forma independiente.

**Gráfico Representativo**

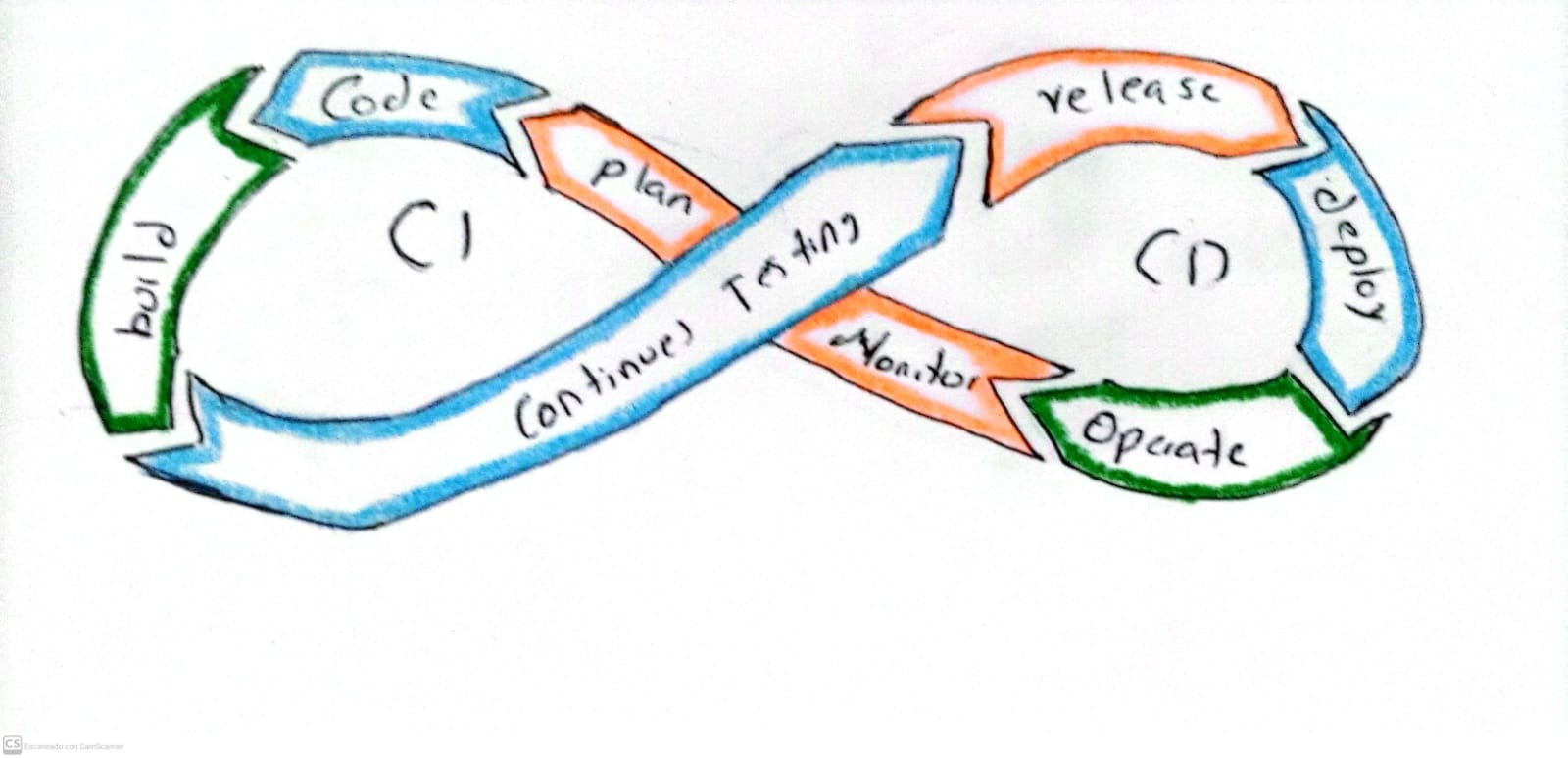
Un gráfico representativo de la evolución hacia los microservicios podría mostrar los siguientes pasos:

1. **Agile Development**
   * Manifiesto Ágil (2001)
   * Iteraciones pequeñas y continuas.
2. **Continuous Integration (CI)**
   * Combina componentes de software temprano.
3. **Continuous Delivery (CD)**
   * Lleva cambios al entorno de producción rápidamente.
4. **DevOps**
   * Fusión de desarrollo y operaciones.
5. **Microservices**
   * Componentes pequeños y desacoplados que colaboran entre sí.

Reflexión:

Una de las reflexiones más importantes de este enfoque es su capacidad para fomentar la colaboración y la flexibilidad. Al descomponer aplicaciones monolíticas en componentes pequeños y desacoplados, los equipos pueden trabajar de manera más autónoma y eficiente, minimizando los cuellos de botella que tradicionalmente obstaculizaban la integración entre desarrollo y operaciones. Este modelo no solo impulsa la velocidad y la escalabilidad, sino que también se alinea con la cultura DevOps, donde la comunicación y la colaboración son fundamentales.

Sin embargo, es crucial recordar que la adopción de microservicios no es una panacea. Requiere un cambio cultural en las organizaciones y una inversión significativa en infraestructura y habilidades. Por lo tanto, la elección de esta arquitectura debe basarse en una evaluación cuidadosa de los requisitos específicos del proyecto y del contexto organizacional.



**Monitoring and Analysis of Microservices Performance**

ste artículo compara las arquitecturas monolíticas y de microservicios en términos de su impacto en la calidad del software. Analiza seis atributos de calidad: acoplamiento, capacidad de prueba, seguridad, complejidad, capacidad de implementación y disponibilidad.

Los autores desarrollaron un modelo matemático utilizando la programación de objetivos enteros mixtos (MIGP) para encontrar la arquitectura óptima en función de estos atributos de calidad. También crearon una aplicación, MicroMono, para demostrar la implementación de su modelo.

El artículo destaca:

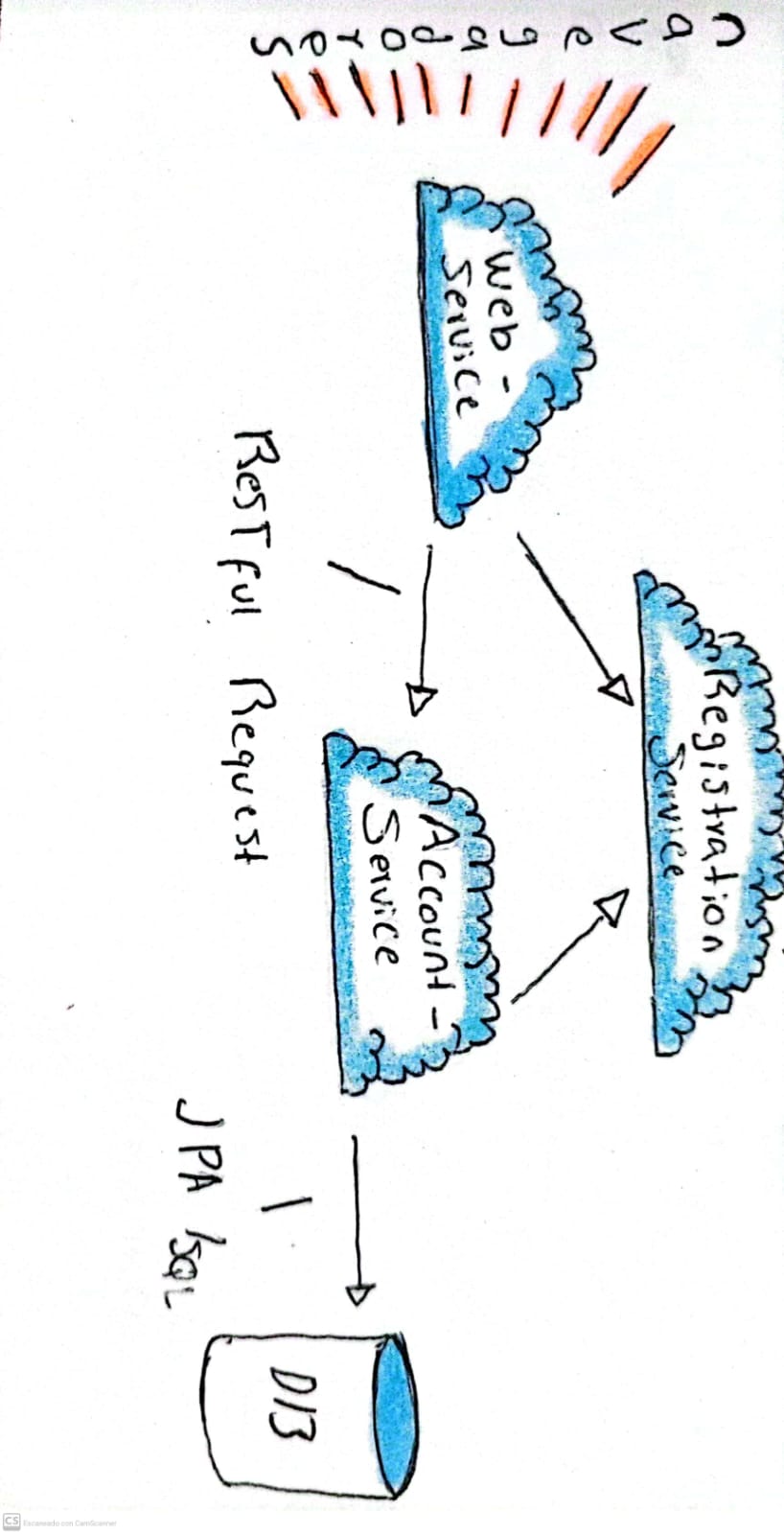
* Un "punto de intersección" donde las arquitecturas monolíticas y de microservicios tienen métricas de calidad de software idénticas, que ocurre solo cuando hay una característica.
* Compensaciones entre arquitecturas, donde los microservicios pueden mejorar la capacidad de implementación y la disponibilidad, pero impactan negativamente en la capacidad de prueba, el acoplamiento y la complejidad.
* Recomendaciones para elegir la mejor arquitectura en función de la importancia de atributos de calidad específicos.
* La importancia de la evaluación continua de la calidad durante todo el proceso de desarrollo de software.

Los autores utilizaron una aplicación basada en microservicios existente, que consta de tres servicios (servicio web, servicio de cuenta y servicio de registro), para probar su modelo. Esta aplicación fue monitoreada y analizada utilizando el marco de trabajo Kieker.

En última instancia, los autores concluyen que la mejor opción entre arquitecturas monolíticas y de microservicios depende de los requisitos específicos y del contexto del sistema de software. Su investigación enfatiza la importancia de un enfoque orientado a la calidad para el desarrollo de software.

Reflexión:

La comparación entre arquitecturas monolíticas y de microservicios subraya la importancia de elegir la mejor opción según las necesidades específicas de un proyecto. Este artículo destaca cómo los microservicios pueden mejorar la implementación y la disponibilidad, aunque también presentan desafíos en acoplamiento y complejidad. La creación de un modelo matemático para evaluar estas arquitecturas permite a los desarrolladores tomar decisiones informadas, enfatizando que un enfoque orientado a la calidad es crucial para el éxito del software. Así, la elección entre ambas arquitecturas debe considerar cuidadosamente el contexto y los requisitos del sistema.

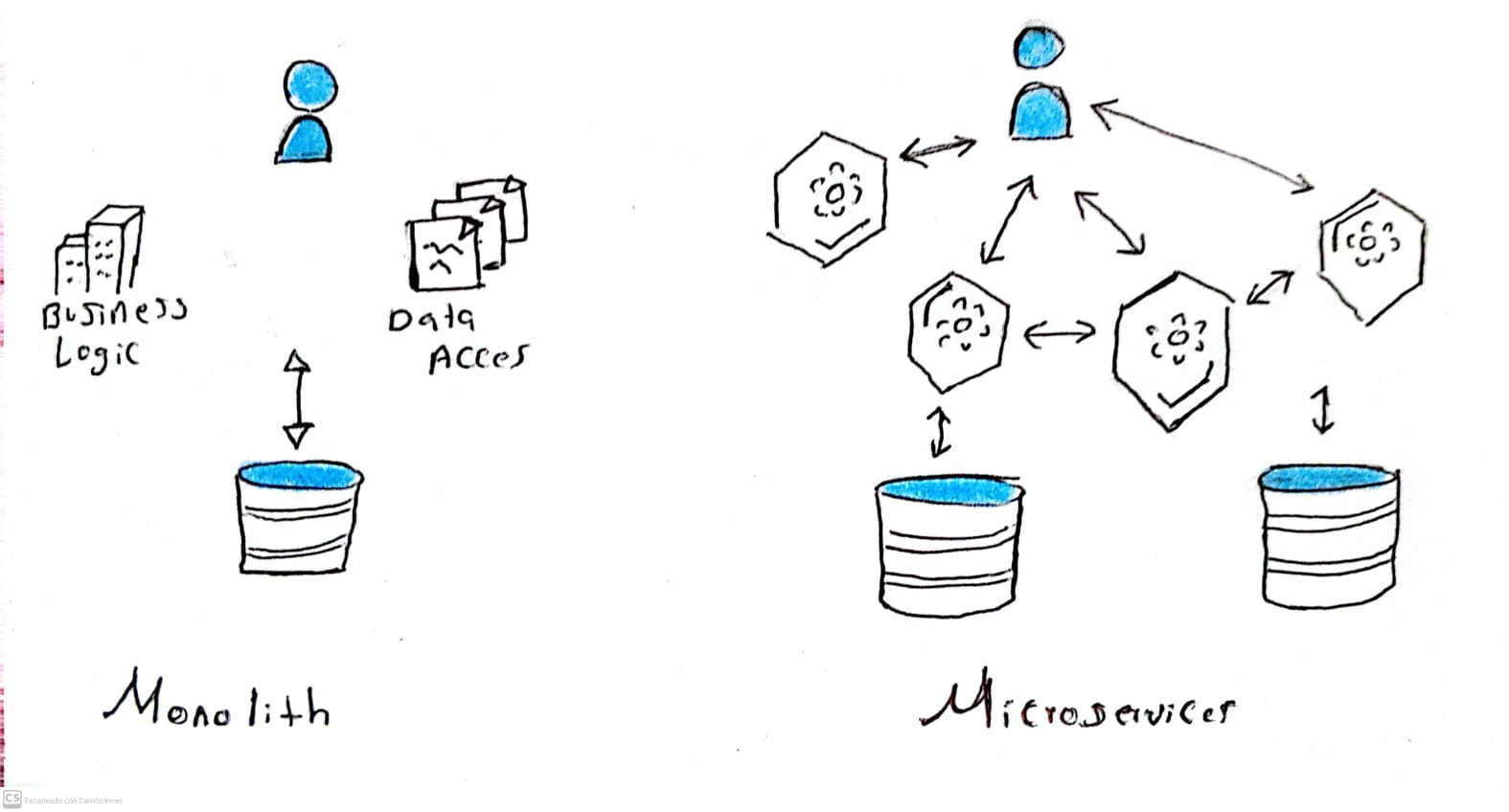


**From Monolithic Systems to Microservices: A Comparative Study of Performance**

El artículo aborda la transición desde los sistemas monolíticos hacia los microservicios, explorando las ventajas y desventajas de ambos enfoques y los retos que surgen en este proceso. Se realiza un estudio de caso que compara el desempeño de una aplicación web implementada en una arquitectura monolítica con otra en una arquitectura de microservicios. Los resultados evidencian que los microservicios presentan una mayor eficiencia en términos de consumo de CPU, utilización de memoria y velocidad de escritura en disco. Sin embargo, se identifican ciertas desventajas en los microservicios, como la complejidad de la gestión y una mayor probabilidad de fallos en las comunicaciones. Los autores concluyen que la arquitectura de microservicios es una opción viable para las aplicaciones que demandan alta escalabilidad y disponibilidad, pero que la arquitectura monolítica puede ser una alternativa más adecuada para aplicaciones de menor complejidad.

Reflexión:

El artículo destaca la transición de arquitecturas monolíticas a microservicios, subrayando que, aunque los microservicios ofrecen mejoras en eficiencia y escalabilidad, también introducen desafíos como la complejidad de gestión y fallos en la comunicación. Esto resalta la importancia de evaluar las necesidades específicas de cada proyecto. Mientras que los microservicios son ideales para aplicaciones de alta demanda, la arquitectura monolítica puede ser más adecuada para sistemas menos complejos. La elección debe basarse en el contexto y los objetivos del software, enfatizando que no hay una solución única para todos los casos.



**Software Architectures - Present and Visions**

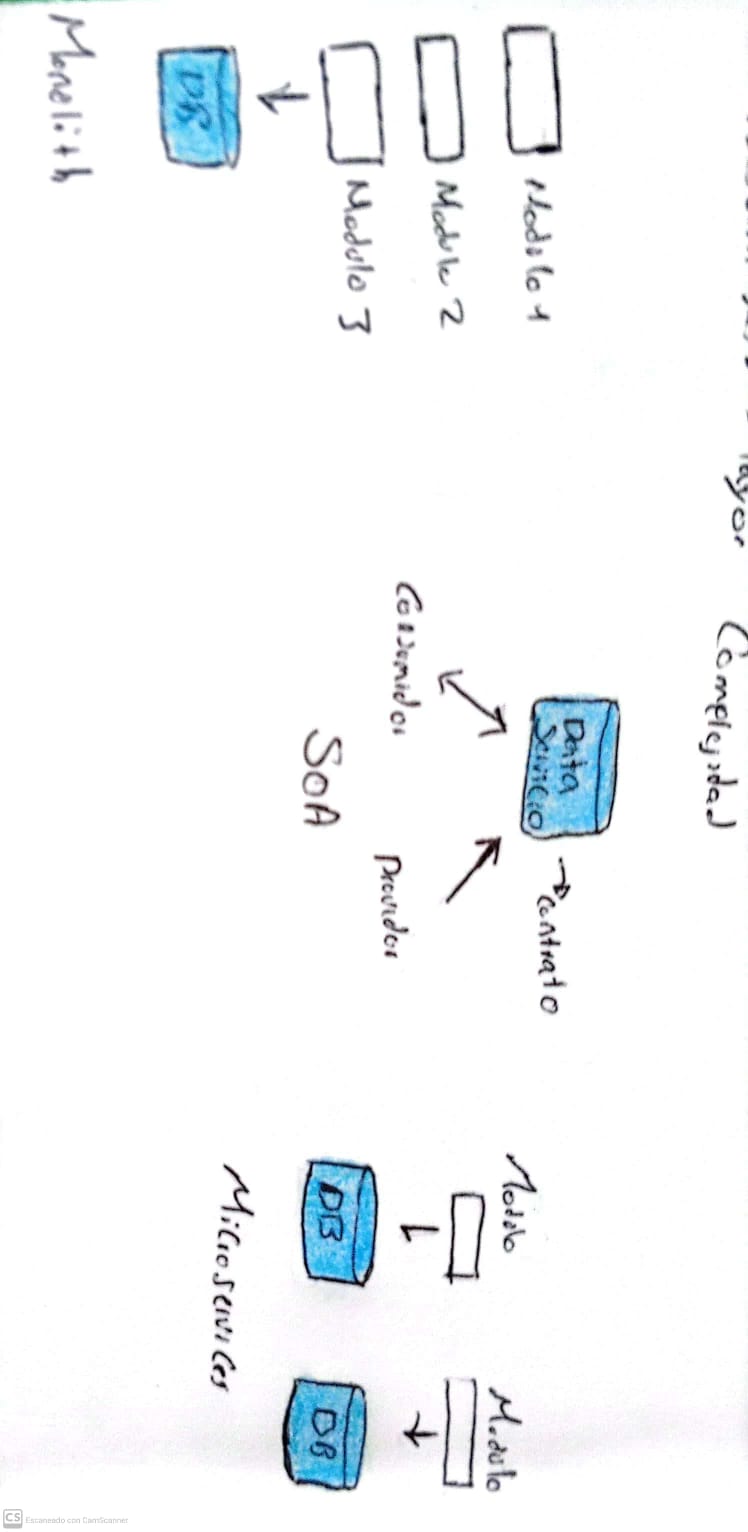
El texto realiza una comparación detallada entre las arquitecturas monolíticas, orientadas a servicios (SOA) y de microservicios, con un enfoque especial en su aplicación en el ámbito universitario. Se destacan las siguientes ventajas y desventajas de cada arquitectura:

* **Arquitectura monolítica:**
  + Ventajas: Simple, fácil de entender y mantener.
  + Desventajas: Dificultad para escalar, actualizar y modificar. Poco flexible y adaptable a cambios.
* **Arquitectura orientada a servicios (SOA):**
  + Ventajas: Mayor modularidad, reutilización de servicios y flexibilidad.
  + Desventajas: Mayor complejidad, requiere una gestión cuidadosa de los servicios y sus interacciones.
* **Arquitectura de microservicios:**
  + Ventajas: Alta escalabilidad, independencia de los servicios, fácil despliegue y actualización.
  + Desventajas: Mayor complejidad de gestión y coordinación entre servicios.

En el contexto universitario, el texto analiza las características de cada arquitectura para abordar las necesidades específicas de sistemas de gestión de aprendizaje, evaluación y administración. Se identifican desafíos y oportunidades que cada arquitectura presenta en este ámbit.

Reflexión:

**Reflexión:** Este análisis revela que la elección de una arquitectura debe ajustarse a las necesidades específicas de la institución universitaria. Cada opción presenta un equilibrio entre simplicidad y escalabilidad, con SOA y microservicios ofreciendo mayor flexibilidad a costa de una mayor complejidad en la gestión. La clave está en evaluar cuidadosamente los requerimientos de la universidad y su capacidad para gestionar la complejidad asociada a cada tipo de arquitectura.



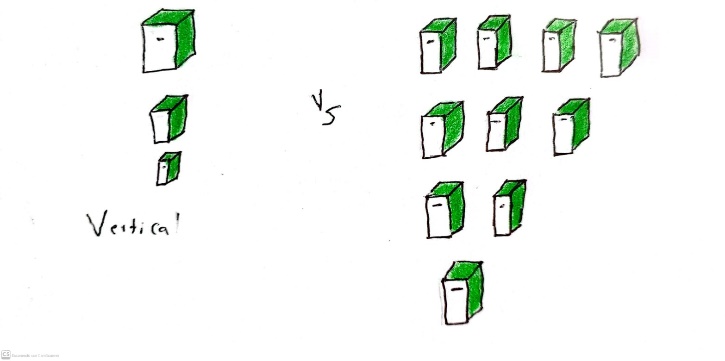
**Development of a Quality-Based Model for Software Architecture Optimization: A Case Study of Monolith and Microservice Architectures**

En este artículo se investigan las ventajas y desventajas entre las arquitecturas monolíticas y de microservicios comparando su rendimiento en un conjunto de atributos de calidad (acoplamiento, capacidad de prueba, seguridad, complejidad, capacidad de implementación y disponibilidad). Los autores determinaron que las arquitecturas monolíticas son mejores para la seguridad, mientras que las arquitecturas de microservicios son mejores para la capacidad de implementación y la disponibilidad. Los autores recomiendan que la elección de una arquitectura se base en los atributos de calidad específicos necesarios para el sistema de software. También señalan que el punto de intersección entre las arquitecturas monolíticas y de microservicios se produce cuando solo hay una característica.

Los autores diseñaron un modelo impulsado por la calidad para la optimización de la arquitectura de software que se puede utilizar para determinar la mejor arquitectura para un conjunto determinado de características. Este modelo utiliza arquitecturas monolíticas y de microservicios y ofrece una guía práctica para los desarrolladores.

Reflexión:

Este artículo enfatiza la importancia de seleccionar la arquitectura adecuada para el desarrollo de software, subrayando que cada opción—monolítica o de microservicios—tiene sus propias ventajas y desventajas. Las arquitecturas monolíticas son más seguras, mientras que los microservicios ofrecen mejor capacidad de implementación y disponibilidad. La recomendación de elegir según las necesidades específicas del sistema resalta un enfoque flexible en el desarrollo. Además, el modelo propuesto ofrece una guía práctica para ayudar a los desarrolladores a tomar decisiones informadas, destacando la relevancia de la calidad en el proceso.



**What are microservices? Your next software architecture**

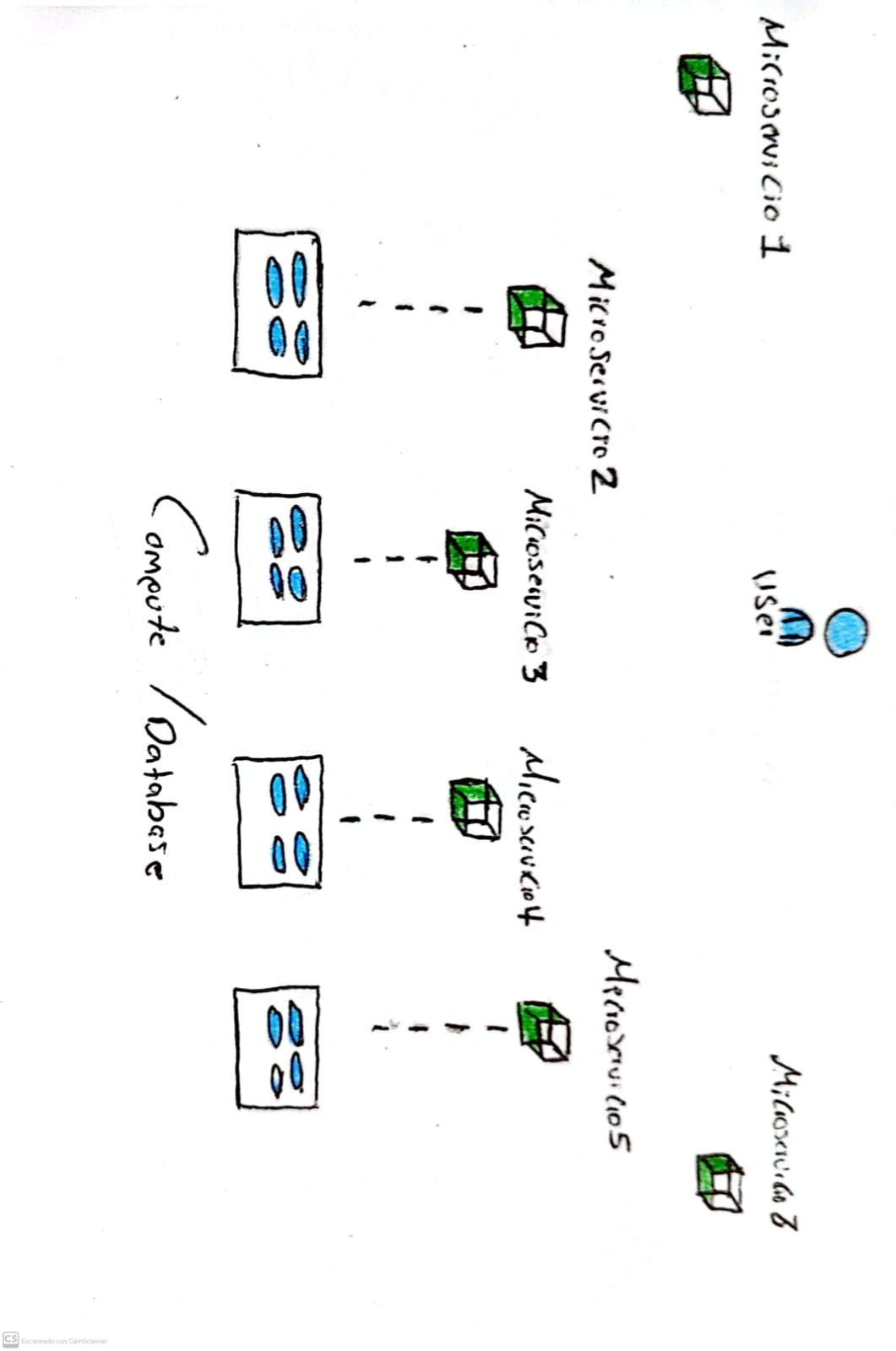
Este artículo explica los microservicios, una arquitectura de software que ha ganado popularidad para crear aplicaciones web modernas. A continuación, se presenta un desglose de los conceptos clave:

* Los microservicios son unidades de código pequeñas e independientes que realizan tareas específicas y se comunican entre sí a través de API, facilitando un desarrollo, escalado y mantenimiento más sencillos.
* Son adecuados para la implementación en la nube, donde los recursos informáticos flexibles optimizan la eficiencia.
* Contrastan con las arquitecturas monolíticas, que agrupan todo el código en una gran aplicación, dificultando el escalado, mantenimiento y actualización.
* Los microservicios son populares en la comunidad Java, con marcos como Spring Boot y Spring Cloud que facilitan su desarrollo.
* Los contenedores, como Docker, permiten una implementación y gestión más eficientes de los microservicios.

El artículo proporciona una comparación entre arquitecturas monolíticas y de microservicios,

Reflexión:

Este artículo destaca cómo los microservicios han revolucionado el desarrollo de aplicaciones web al permitir un enfoque más modular y flexible. Su capacidad para escalar y adaptarse a entornos en la nube, junto con el uso de tecnologías como Docker, facilita la creación de aplicaciones complejas. La comparación con las arquitecturas monolíticas resalta la necesidad de evolucionar en las prácticas de desarrollo para abordar las demandas actuales del mercado.



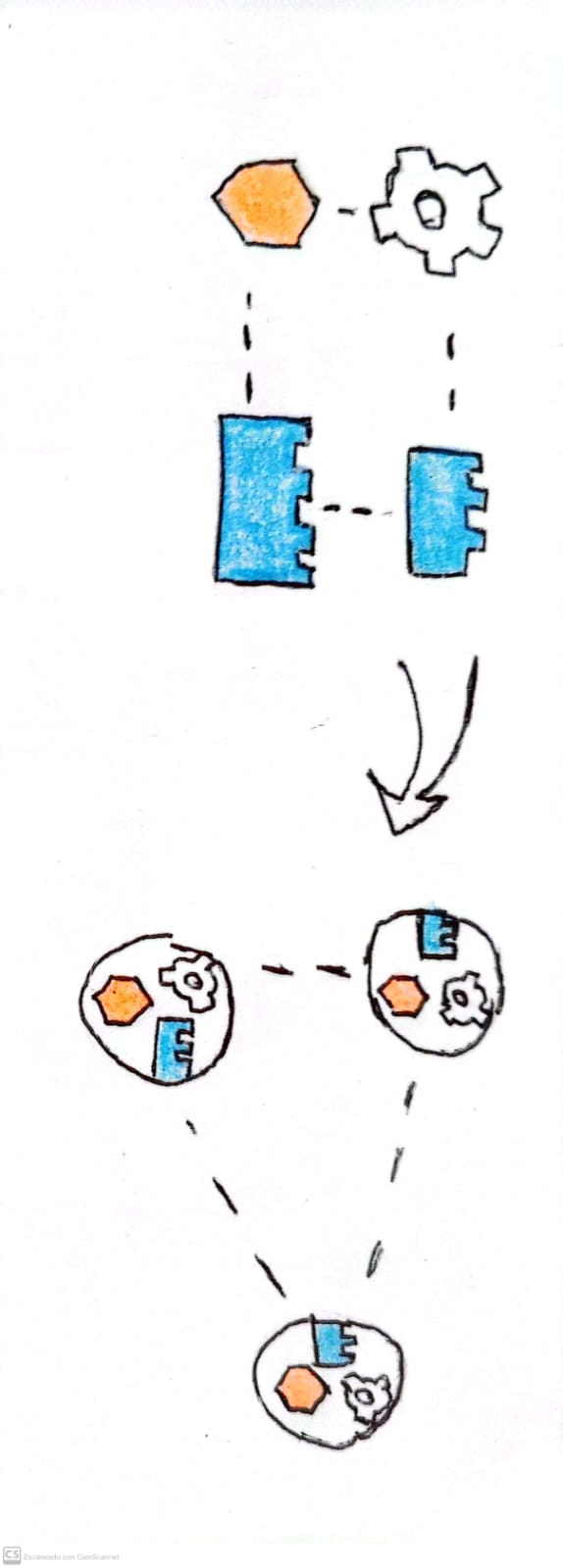
**Evolution styles: foundations and models for software architecture evolution**

Este artículo se centra en la evolución de la arquitectura de software, en particular el cambio de arquitecturas monolíticas a arquitecturas de microservicios. Se explica que las arquitecturas monolíticas agrupan todo el código en una gran aplicación, lo que puede dificultar su gestión, escalabilidad y actualización. En contraste, los microservicios dividen las aplicaciones en pequeñas unidades de código independientes que se comunican a través de API, permitiendo una mayor flexibilidad y escalabilidad.

Los microservicios se alinean con los principios de DevOps, facilitando lanzamientos más rápidos y frecuentes. El artículo presenta un marco para modelar y planificar la evolución arquitectónica, incluyendo rutas de evolución y restricciones. Un estudio de caso sobre la evolución de una aplicación de motor de ajedrez resalta los beneficios y desafíos de este proceso. En general, el artículo sostiene que las arquitecturas de microservicios son prometedoras para construir sistemas de software modernos, especialmente para aplicaciones grandes y complejas, enfatizando la importancia de una planificación cuidadosa al migrar de arquitecturas monolíticas.

Reflexión:

El artículo ilustra claramente la transición de arquitecturas monolíticas a microservicios, resaltando sus ventajas en flexibilidad y escalabilidad. Los microservicios, alineados con DevOps, permiten ciclos de desarrollo más ágiles y adaptativos. Sin embargo, el éxito de esta migración requiere una planificación meticulosa y un análisis de los desafíos potenciales. En resumen, mientras que los microservicios ofrecen un enfoque moderno y prometedor para aplicaciones complejas, cada transición debe ser cuidadosamente considerada en función de las necesidades específicas del proyecto.



# **Micro services: Breaking down software monoliths**

Este artículo aborda la arquitectura de microservicios, un enfoque moderno en el desarrollo de software que descompone aplicaciones grandes y monolíticas en unidades más pequeñas e independientes. Los microservicios son autónomos y realizan tareas específicas, comunicándose entre sí a través de APIs ligeras. Entre sus ventajas se encuentran interrupciones mínimas del servicio, flexibilidad en el uso de lenguajes y plataformas, mayor cohesión y menos acoplamiento, así como una fácil recuperación de datos. El artículo menciona que empresas como Amazon, Netflix, Google, eBay, Uber y Groupon utilizan microservicios, subrayando su popularidad y los beneficios que ofrecen en términos de agilidad y competitividad.

**Reflexión:**

La arquitectura de microservicios representa un cambio significativo en la forma en que se desarrollan y gestionan las aplicaciones. Al permitir que cada componente funcione de manera independiente, se minimizan las interrupciones y se facilita el mantenimiento y la escalabilidad. Este enfoque se adapta perfectamente a las necesidades del mercado actual, donde la agilidad y la capacidad de respuesta son cruciales para el éxito empresarial. La adopción de microservicios por empresas líderes resalta su eficacia y potencial para transformar la industria del software.

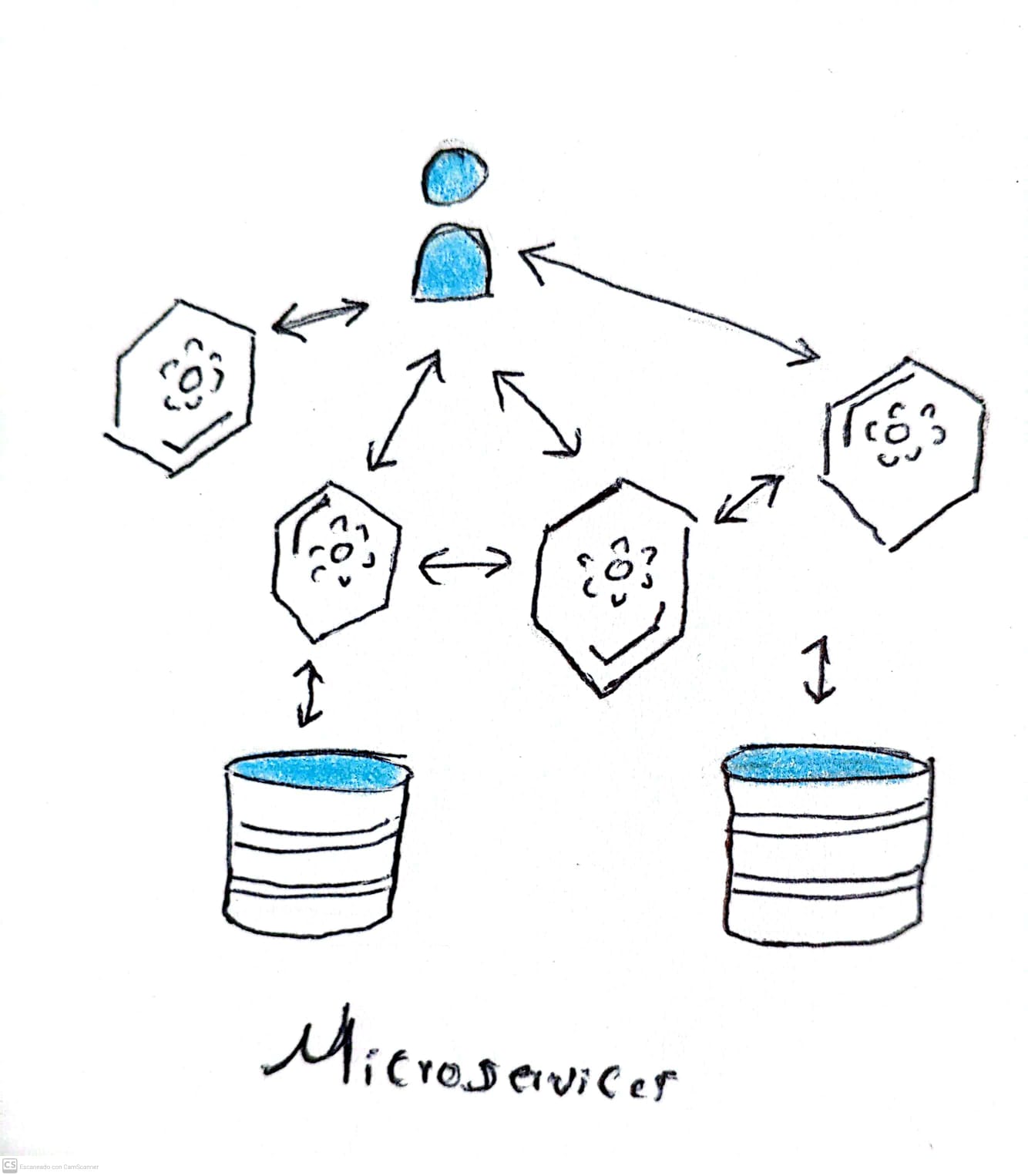


# **A Comprehensive Analysis of the Literature on the "Traditional" Monolithic Approach and Micro Services Architecture**

Este artículo proporciona un análisis exhaustivo sobre las arquitecturas monolíticas tradicionales y la arquitectura de microservicios, que está ganando popularidad en el desarrollo de aplicaciones modernas, escalables y flexibles. Se contrastan ambas arquitecturas, destacando las ventajas de los microservicios, como el desarrollo e implementación independientes de servicios, mejor escalabilidad y agilidad, y un mayor aislamiento de fallas. Los autores subrayan cómo los microservicios benefician a los equipos de desarrollo y a las empresas, promoviendo un trabajo en equipo más ágil, eficiencia y mayor flexibilidad en las opciones tecnológicas. Sin embargo, también se abordan los desafíos asociados con los microservicios, incluyendo la complejidad del diseño, dificultades en la comunicación y preocupaciones de seguridad y gestión de datos. El artículo concluye enfatizando la relevancia de los microservicios en la creación de aplicaciones modernas y sugiere futuras oportunidades de investigación en este ámbito, siendo un recurso valioso para desarrolladores e investigadores interesados en la evolución de la arquitectura de software.

**Reflexión:**

Este análisis resalta cómo la transición de arquitecturas monolíticas a microservicios está moldeando el futuro del desarrollo de software. La capacidad de escalar y adaptar aplicaciones de manera más ágil responde a las demandas actuales del mercado, mientras que los desafíos identificados enfatizan la necesidad de un enfoque cuidadoso en el diseño y la gestión. La popularidad creciente de los microservicios entre empresas líderes subraya su efectividad y su potencial para mejorar la eficiencia y la colaboración. Al mismo tiempo, el artículo abre la puerta a nuevas investigaciones, sugiriendo que aún hay mucho por explorar en esta área.



**PolyGlot Persistence for Microservices-Based Applications**

Este artículo propone un nuevo marco arquitectónico para aplicaciones basadas en microservicios que utiliza persistencia políglota, donde se utilizan diferentes tipos de bases de datos para diferentes módulos de la aplicación. Las principales ventajas de este enfoque son:

* **Funcionalidad mejorada**: el uso de diferentes bases de datos para diferentes módulos permite que las aplicaciones aprovechen las características únicas de cada base de datos, lo que da como resultado aplicaciones más potentes y flexibles.
* **Escalabilidad mejorada**: el uso de microservicios permite escalar de forma independiente cada servicio en función de sus necesidades específicas.
* **Mayor capacidad de mantenimiento**: los microservicios están diseñados para estar acoplados de forma flexible, lo que facilita la actualización y el mantenimiento de servicios individuales.

El artículo describe el marco propuesto en detalle, incluido el diseño de API, la contenedorización mediante Docker y el proceso de implementación. Los autores también validan su enfoque comparándolo con modelos existentes utilizando el estándar ISO/IEC 25010. Los resultados muestran que el marco propuesto supera a los enfoques tradicionales en términos de portabilidad, capacidad de mantenimiento y rendimiento. El artículo concluye abogando por la adopción generalizada de la persistencia políglota en aplicaciones basadas en microservicios, destacando los beneficios para las aplicaciones modernas y complejas que requieren una gestión de datos flexible y escalable.

El artículo también pide más investigación en esta área para explorar el uso de una gama más amplia de bases de datos heterogéneas y desarrollar herramientas y métodos más sofisticados para diseñar y evaluar arquitecturas de microservicios persistentes políglotas.

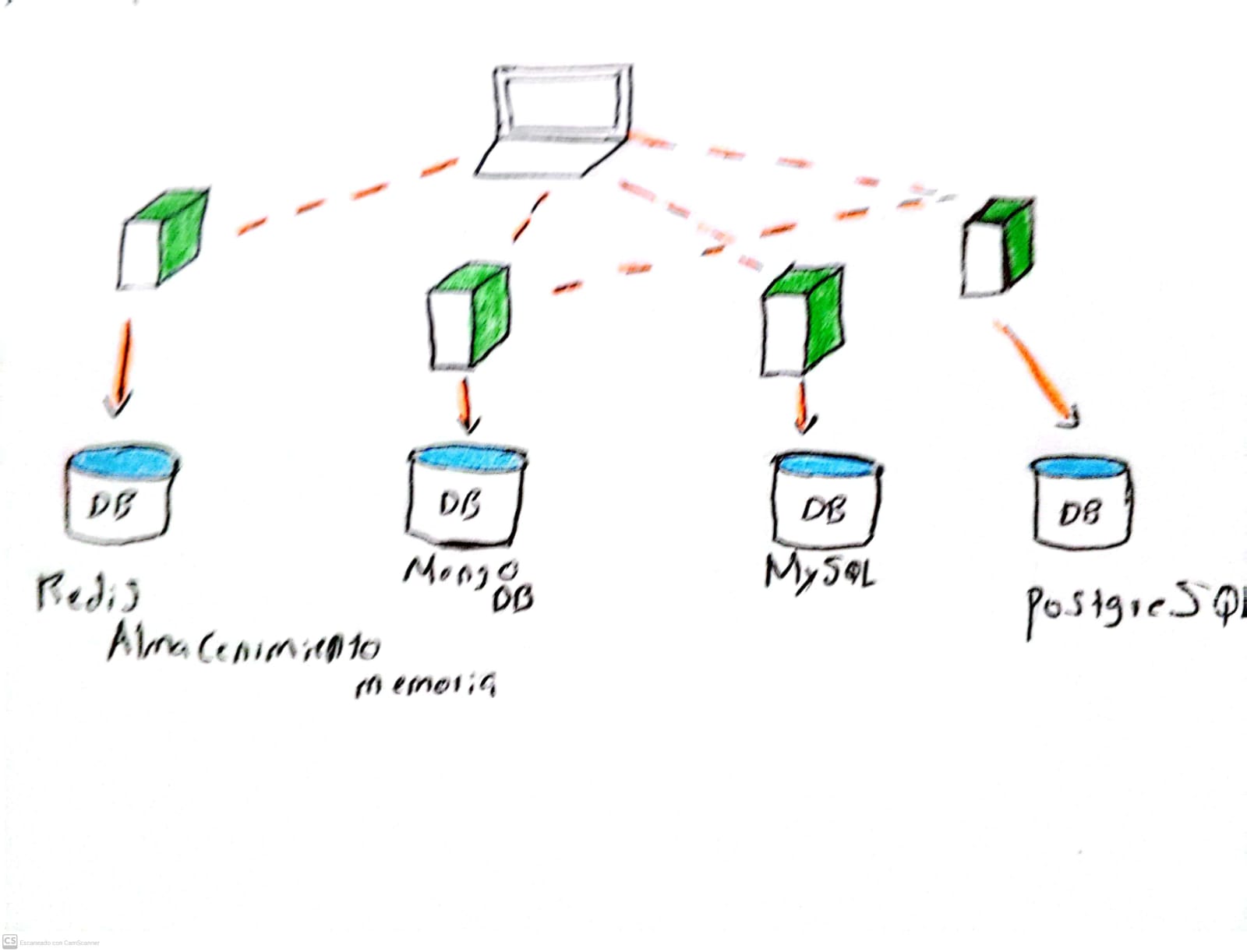
A continuación, se incluye un resumen simplificado de los hallazgos clave:

* La persistencia políglota es un enfoque útil para las aplicaciones de microservicios, ya que permite a los desarrolladores aprovechar las fortalezas de diferentes bases de datos.
* El marco arquitectónico propuesto para la persistencia políglota ofrece una escalabilidad, capacidad de mantenimiento y rendimiento mejorados.
* Los autores sugieren que el trabajo futuro debería centrarse en el uso de una gama más amplia de bases de datos heterogéneas y el desarrollo de herramientas y métodos más sofisticados para arquitecturas de microservicios persistentes políglotas.

Este artículo proporciona información valiosa para los desarrolladores y arquitectos que están considerando el uso de la persistencia políglota en sus aplicaciones basadas en microservicios.

Reflexión:

La propuesta de un marco arquitectónico para aplicaciones basadas en microservicios con persistencia políglota destaca la importancia de adaptar las tecnologías de datos a las necesidades específicas de cada módulo. Este enfoque no solo mejora la funcionalidad y la escalabilidad, sino que también facilita el mantenimiento de sistemas complejos. A medida que las aplicaciones se vuelven más interdependientes, la capacidad de elegir la base de datos adecuada para cada servicio se convierte en un activo valioso. La investigación y el desarrollo en este ámbito son esenciales para maximizar los beneficios de la persistencia políglota, impulsando así la innovación y la eficiencia en el desarrollo de software moderno.



# **Microservices Identification in Monolith Systems: Functionality Redesign Complexity and Evaluation of Similarity Measures**

Este artículo investiga la precisión y el esfuerzo que implica el uso de métodos de evaluación del rendimiento monolíticos y basados en componentes para arquitecturas de software, centrándose en los desafíos prácticos de migrar de sistemas monolíticos a sistemas de microservicios. El estudio concluye que:

* **Los métodos monolíticos** (como SPE, umlPSI y Capacity Planning) pueden proporcionar predicciones de rendimiento razonablemente precisas, pero requieren un esfuerzo significativo para crear modelos.
* **Los métodos basados en componentes** (como PCM) ofrecen una alternativa prometedora, que logra una precisión similar pero con el potencial de ahorrar tiempo y esfuerzo mediante la reutilización de modelos de componentes.
* Reutilizar modelos de componentes es significativamente más rápido que crearlos desde cero, y el esfuerzo requerido para crear nuevos modelos está influenciado por la complejidad de la lógica interna del componente y sus interacciones con otros componentes.

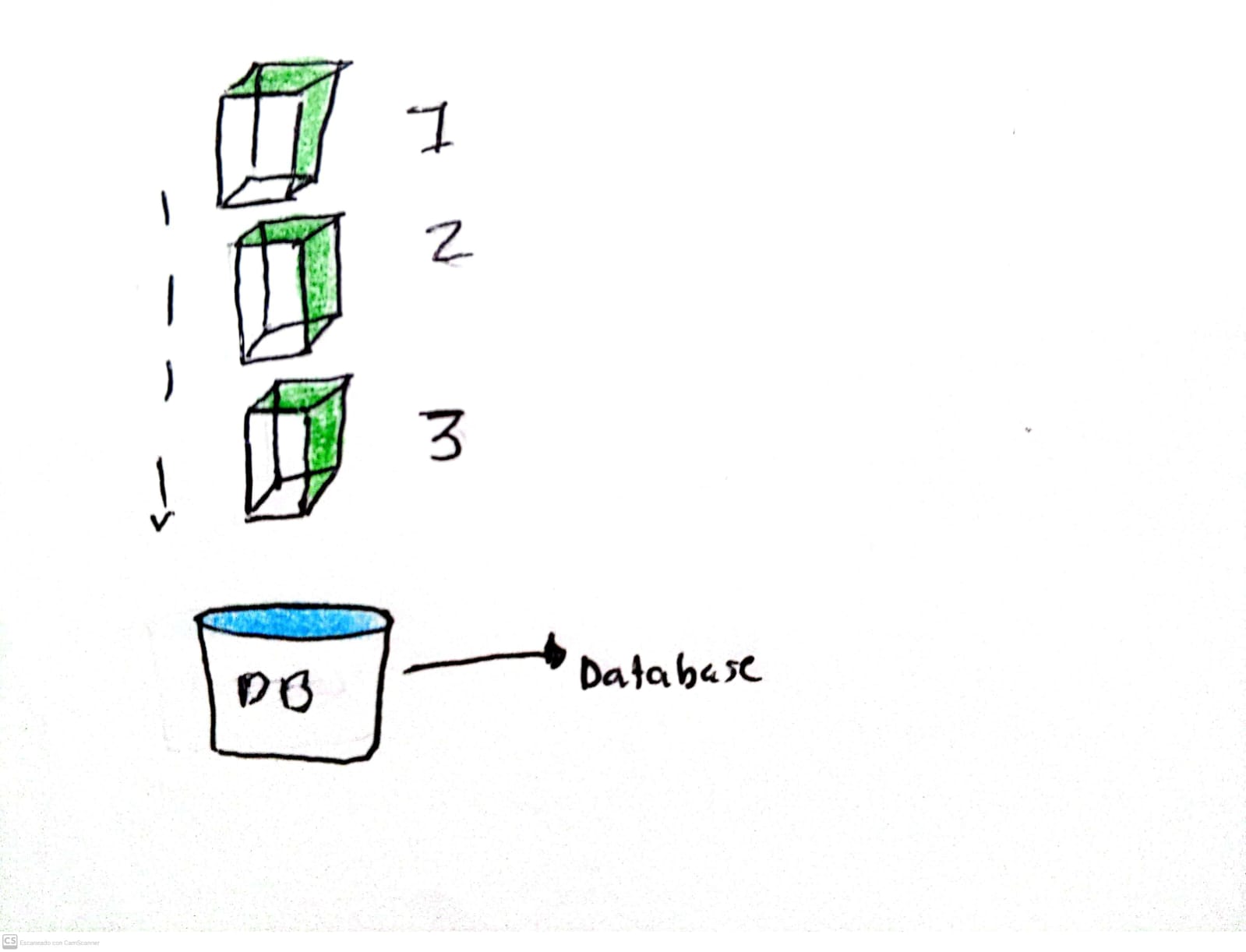
El artículo sugiere que la elección entre enfoques monolíticos y basados en componentes depende de las necesidades específicas del proyecto y la experiencia del equipo de desarrollo. Destaca la necesidad de realizar más investigaciones sobre:

* Mejorar la aplicabilidad de los métodos basados en componentes en entornos industriales.
* Desarrollar herramientas para automatizar los procesos de creación y validación de modelos.
* Comprender las ventajas y desventajas que implica elegir diferentes niveles de abstracción para los modelos de componentes.
* Realizar análisis de costo-beneficio para evaluar el valor de diferentes enfoques de evaluación del rendimiento.

Los autores enfatizan que la transición hacia métodos basados en componentes y el desarrollo adicional de herramientas robustas son importantes para facilitar la adopción exitosa de la arquitectura de microservicios en entornos industriales. Esta investigación ofrece información valiosa para desarrolladores y arquitectos que estén considerando la migración de arquitecturas monolíticas a arquitecturas de microservicios.

Reflexion:

El artículo subraya la importancia de evaluar cuidadosamente los métodos de rendimiento al migrar de arquitecturas monolíticas a microservicios. Mientras que los enfoques monolíticos pueden ofrecer predicciones precisas, su complejidad y el esfuerzo involucrado en la creación de modelos son consideraciones críticas. Por otro lado, los métodos basados en componentes se presentan como una alternativa valiosa, permitiendo la reutilización y ahorro de tiempo. Esta investigación no solo destaca la necesidad de adaptar las herramientas y métodos a las necesidades específicas de cada proyecto, sino también la urgencia de seguir explorando y desarrollando soluciones que faciliten esta transición, impulsando así la innovación y la eficiencia en el desarrollo de software.



# **Monolith to Microservices: Representing Application Software through Heterogeneous Graph Neural Network**

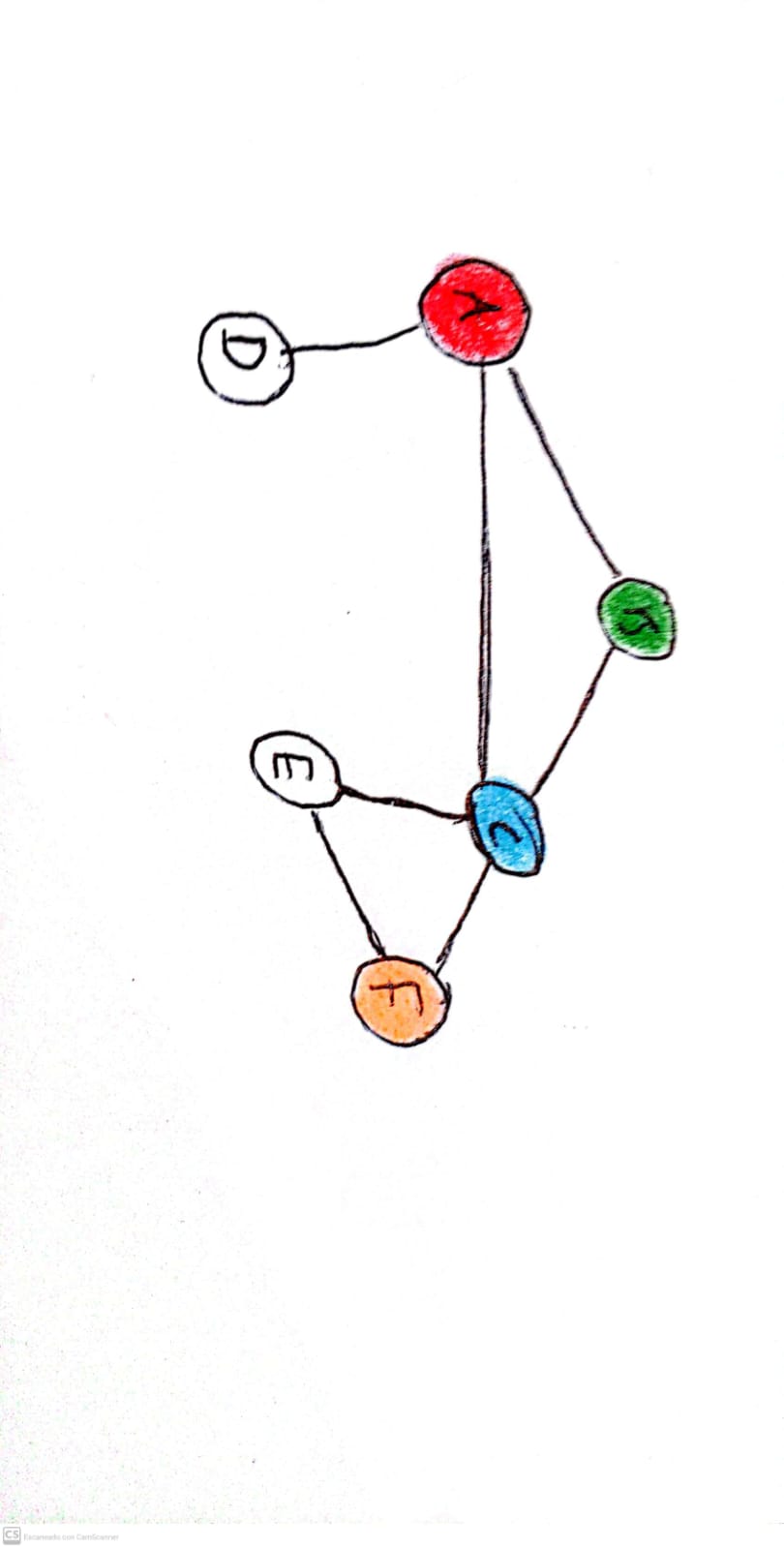
Este artículo propone un nuevo enfoque para descomponer aplicaciones monolíticas en microservicios utilizando una red neuronal de grafos heterogéneos (GNN). A continuación, se ofrece un desglose simple:

* **Las arquitecturas monolíticas** son tradicionales, pero se vuelven difíciles de mantener y escalar a medida que crecen.
* **Los microservicios** ofrecen una alternativa más flexible y escalable, pero migrar de una arquitectura monolítica a una de microservicios puede ser un desafío.
* **Los autores proponen un enfoque de aprendizaje de representación** que utiliza un grafo heterogéneo para representar tanto artefactos de software (programas y recursos) como sus relaciones (llamadas de función, herencia, etc.).
* **Su novedosa CHGNN** (red neuronal de grafos heterogéneos con conocimiento de la comunidad) realiza una agrupación basada en restricciones en este grafo, teniendo en cuenta los conjuntos de semillas definidos por expertos para guiar el proceso.
* **Los autores evalúan su enfoque** utilizando cuatro aplicaciones monolíticas disponibles públicamente y lo comparan con los enfoques existentes, demostrando que CHGNN supera sistemáticamente a otros métodos.

La investigación destaca el potencial de las GNN para la descomposición de microservicios y enfatiza la necesidad de considerar tanto la información estructural como la de comportamiento, junto con las restricciones definidas por los expertos, para obtener resultados más precisos y efectivos.

Reflexión:

El artículo resalta la relevancia de utilizar redes neuronales de grafos heterogéneos (GNN) para la descomposición de aplicaciones monolíticas en microservicios. A medida que las arquitecturas monolíticas enfrentan desafíos de escalabilidad y mantenimiento, este enfoque innovador proporciona una solución prometedora al representar artefactos de software y sus relaciones en un grafo. La implementación de la CHGNN, que incorpora conocimiento experto, demuestra que se pueden lograr descomposiciones más precisas y efectivas al considerar tanto la estructura como el comportamiento del software. Esta investigación no solo abre nuevas posibilidades en la migración a microservicios, sino que también enfatiza la importancia de integrar técnicas avanzadas de aprendizaje automático en el desarrollo de software moderno.



# **A New Decomposition Method for Designing Microservices**

Este artículo presenta un nuevo método para descomponer aplicaciones monolíticas en microservicios, una arquitectura popular para aplicaciones basadas en la nube. A continuación, se ofrece una explicación simplificada de los puntos clave:

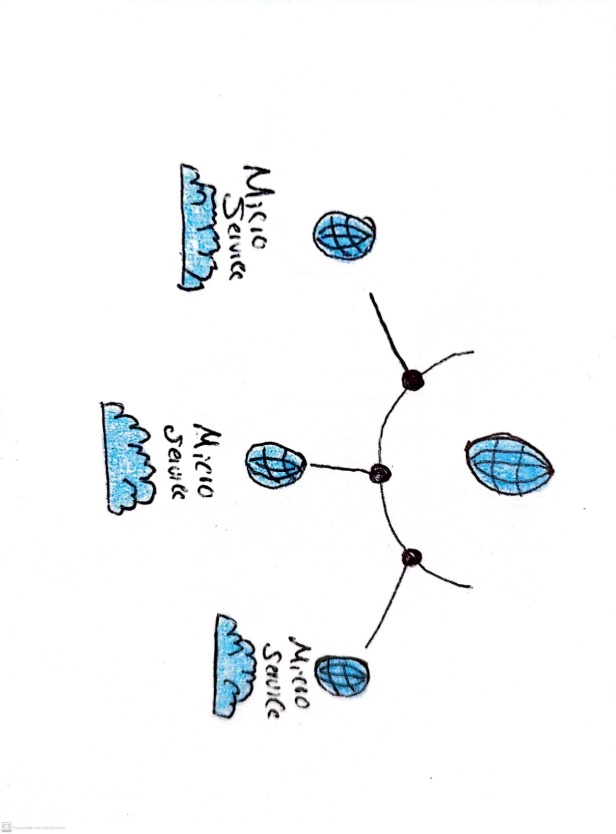
* **Los microservicios** son unidades de código más pequeñas e independientes que se comunican a través de API. Esta arquitectura permite aplicaciones más flexibles y escalables en comparación con los sistemas monolíticos tradicionales.
* **El artículo propone un método** para ayudar a los arquitectos de software a identificar microservicios adecuados a partir de aplicaciones monolíticas mediante el análisis de sus especificaciones de API.
* **El método implica tres pasos**:
  1. Extracción de nombres de operaciones de las especificaciones OpenAPI.
  2. Conversión de nombres de operaciones en incrustaciones de palabras utilizando el modelo fastText.
  3. Agrupamiento de nombres de operaciones similares utilizando el algoritmo Affinity Propagation.

Los autores probaron su método utilizando especificaciones OpenAPI de cuatro aplicaciones diferentes: Amazon Web Services, PayPal, Kanban Board y Money Transfer. Los resultados mostraron un rendimiento prometedor, logrando una medida F de 0,81, una precisión de 0,84 y una recuperación de 0,78.

El artículo concluye que este método es una herramienta valiosa para los arquitectos de software que buscan migrar aplicaciones monolíticas a microservicios. También identifica futuras direcciones de investigación, como el uso de diferentes métodos de cálculo de distancia para vectores de palabras y el desarrollo de nuevas métricas de evaluación que consideren la granularidad del servicio.

Reflexión:

El artículo destaca la creciente necesidad de descomponer aplicaciones monolíticas en microservicios para mejorar la flexibilidad y escalabilidad de las arquitecturas de software. Al proponer un método basado en el análisis de especificaciones OpenAPI, ofrece una herramienta práctica para los arquitectos de software en la migración a microservicios. Los resultados prometedores demuestran la efectividad del enfoque, subrayando la importancia de utilizar tecnologías avanzadas como fastText y algoritmos de agrupamiento. Esta investigación no solo contribuye a la evolución de la arquitectura de software, sino que también abre la puerta a futuras exploraciones en la optimización de procesos de migración, lo que es crucial en un entorno tecnológico en constante cambio.

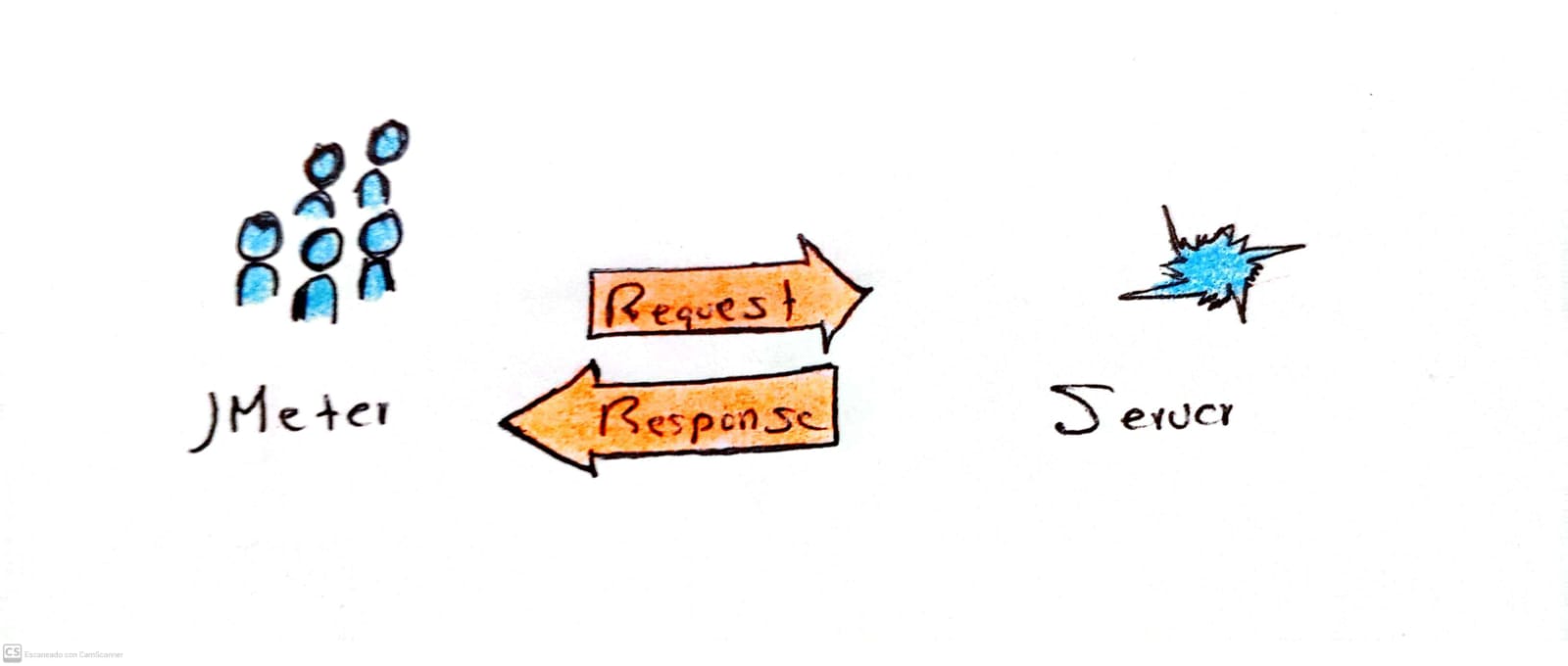


**Arquitectura basada en Microservicios y DevOps para una ingeniería de software continua**

Los microservicios son una arquitectura que utiliza servicios independientes y escalables para desarrollar aplicaciones complejas, mientras que DevOps es un enfoque que promueve la integración y entrega continua de software, reduciendo la brecha entre desarrollo y operaciones. Este trabajo propone una arquitectura que combina microservicios y DevOps para una ingeniería de software continua. Se aplica en un caso de estudio con estudiantes, resultando en un conjunto de aplicaciones implementadas con tecnologías innovadoras.

Reflexión:

a combinación de microservicios y DevOps transforma el desarrollo de aplicaciones complejas al permitir la creación de servicios independientes y escalables, junto con una integración y entrega continua que mejora la colaboración entre desarrollo y operaciones. Este enfoque reduce los tiempos de entrega y mejora la calidad del software. El caso de estudio con estudiantes demuestra cómo estas tecnologías innovadoras pueden aplicarse en la educación, preparándolos para un entorno laboral ágil y colaborativo, fundamental en la ingeniería de software actual.

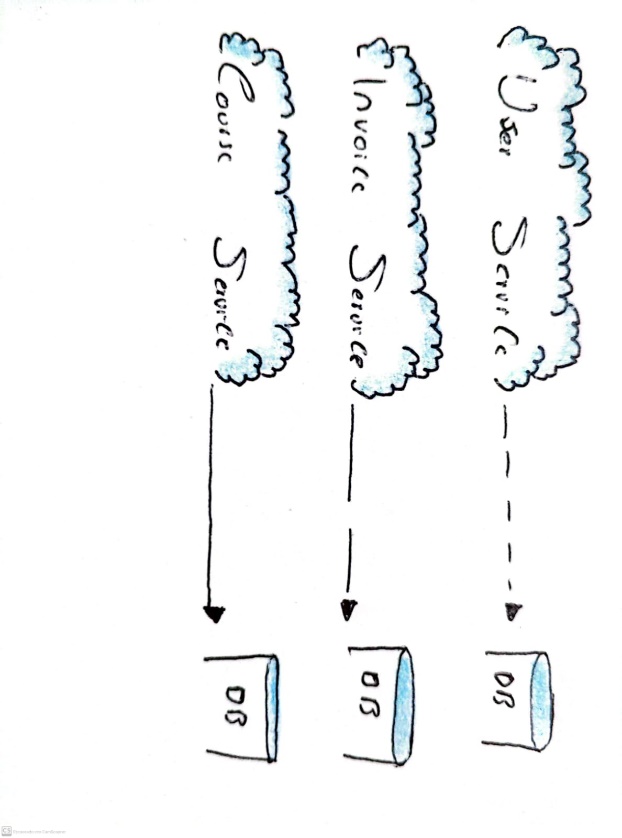


# **Assessing the Impact of Migration from SOA to Microservices Architecture**

El Artículo explora la transición de aplicaciones basadas en SOA (Arquitectura Orientada a Servicios) a microservicios, un estilo arquitectónico adoptado por empresas como Netflix y Twitter. Aunque el enfoque de microservicios presenta ventajas significativas, como la escalabilidad y flexibilidad, también plantea desafíos relacionados con el rendimiento y la complejidad durante la migración. El estudio utiliza métricas como la probabilidad de propagación de cambios y la estabilidad arquitectónica para evaluar el impacto de esta migración. Los resultados muestran que, a pesar de los desafíos, los microservicios ofrecen beneficios sustanciales y son especialmente adecuados para aplicaciones empresariales grandes.

Reflexión:

El artículo destaca la evolución de las aplicaciones de SOA a microservicios, un cambio adoptado por líderes como Netflix y Twitter. Aunque los microservicios ofrecen ventajas claras en escalabilidad y flexibilidad, también conllevan retos en rendimiento y complejidad durante la migración. Al evaluar esta transición mediante métricas específicas, los resultados indican que, a pesar de los desafíos, los beneficios de los microservicios son significativos, haciéndolos particularmente adecuados para grandes aplicaciones empresariales. Esta reflexión subraya la importancia de sopesar tanto los pros como los contras al considerar una migración arquitectónica.

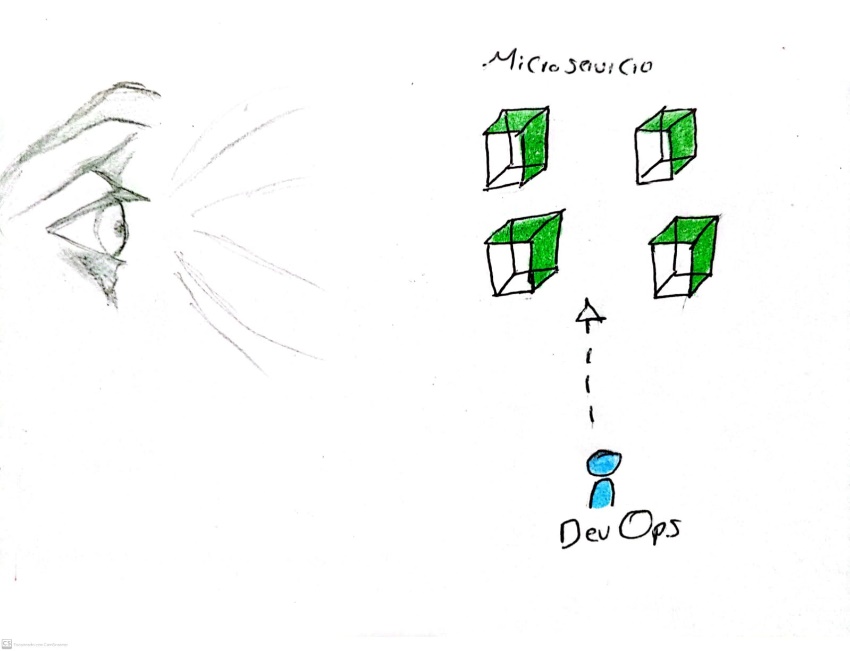


**Optimized Strategy for Inter-Service Communication in Microservices**

Este artículo trata sobre la migración de aplicaciones monolíticas a la arquitectura de microservicios en la nube. Aunque la nube está diseñada para manejar artefactos ligeros como microservicios, simplemente trasladar software monolítico a la nube no aprovecha al máximo sus beneficios. La arquitectura monolítica, al estar estrechamente acoplada, presenta desafíos para adaptarse a las cambiantes necesidades de los usuarios en términos de escalabilidad y flexibilidad. Los microservicios, en cambio, permiten un diseño distribuido y desacoplado, facilitando el mantenimiento y la agilidad empresarial, aunque con el inconveniente de mayor latencia en la comunicación entre servicios.  
  
El estudio propone una solución para reducir la latencia en la comunicación inter-servicios utilizando un patrón de comunicación asíncrona con Redis Stream, mejorando el rendimiento global.

Reflexión:

La migración de aplicaciones monolíticas a microservicios en la nube es un cambio significativo, pero simplemente trasladar un monolito no maximiza los beneficios de la nube. La arquitectura monolítica limita la escalabilidad y flexibilidad, mientras que los microservicios ofrecen un diseño desacoplado que mejora el mantenimiento y la agilidad. Sin embargo, presentan desafíos como la latencia en la comunicación entre servicios. La propuesta de usar comunicación asíncrona con Redis Stream para reducir esta latencia es un enfoque prometedor que puede optimizar el rendimiento general. Esto resalta la necesidad de una migración estratégica que aproveche plenamente las capacidades de la nube.



# **Designing Online Healthcare Using DDD in Microservices Architecture**

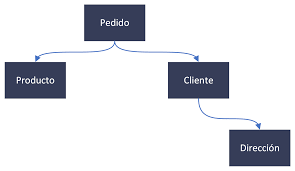
Este estudio aborda el diseño de un sistema de atención médica en línea utilizando una arquitectura de microservicios con un enfoque de Diseño Dirigido por el Dominio (DDD). La transición de una arquitectura monolítica a microservicios responde a la necesidad de agilidad, flexibilidad y seguridad en el mantenimiento de software. Esta transformación promueve la independencia entre módulos, facilita la integración de tecnologías diversas, y reduce el tiempo de desarrollo.

El proceso de diseño propuesto se basa en analizar los procesos comerciales del sistema actual, identificar los requisitos del sistema y luego rediseñarlo con microservicios. Al aplicar DDD, se asegura que el sistema esté bien alineado con el dominio del sector salud, el cual incluye una vasta cantidad de entidades e intercambios de información, como pacientes, médicos, hospitales y aseguradoras.

Reflexión:

Este estudio enfatiza la transición de una arquitectura monolítica a microservicios en el diseño de un sistema de atención médica en línea, destacando su capacidad para mejorar la agilidad y flexibilidad del software. Al aplicar un enfoque de Diseño Dirigido por el Dominio (DDD), se asegura que el sistema esté alineado con las necesidades del sector salud, optimizando la interacción entre entidades como pacientes, médicos y aseguradoras.

Esta transformación promueve la independencia de módulos, facilita la integración de nuevas tecnologías y reduce los tiempos de desarrollo. En un sector tan dinámico como la atención médica, esta capacidad de adaptación es crucial para ofrecer servicios de calidad y responder a las expectativas de los usuarios. En resumen, el uso de microservicios y DDD puede revolucionar la atención médica en línea, creando sistemas más robustos y centrados en el usuario.



**The Comparison of Microservice and Monolithic Building Enterprise Architecture Agility and Sustenance with SOA**

La Arquitectura Orientada a Servicios (SOA) es un marco técnico diseñado para construir aplicaciones con acoplamiento promoviendo la flexibilidad y agilidad empresarial. Facilita el diseño e implementación simultáneos de procesos empresariales mediante una estructura jerárquica de servicios. La adopción de SOA ha sido obstaculizada por malentendidos sobre su naturaleza. Este artículo revisa los orígenes históricos de SOA, sus bases tecnológicas y su integración con plataformas actuales, enfatizando la reutilización y el acoplamiento flexible. También explora las metodologías de diseño, las plataformas y la integración de aplicaciones de SOA, y finalmente discute los desafíos y direcciones futuras de investigación para comprender mejor su potencial técnico y empresarial.

Reflexión:

La Arquitectura Orientada a Servicios (SOA) se presenta como un marco esencial para la construcción de aplicaciones que requieren flexibilidad y agilidad empresarial. Su estructura jerárquica de servicios permite un diseño e implementación simultáneos, facilitando la adaptación a las dinámicas cambiantes del mercado. Sin embargo, su adopción se ha visto limitada por malentendidos que han distorsionado su verdadera naturaleza y potencial.

Este artículo no solo revisa los orígenes históricos y las bases tecnológicas de SOA, sino que también destaca su integración con plataformas contemporáneas, lo que subraya la importancia de la reutilización y el acoplamiento flexible. Al explorar metodologías de diseño y plataformas de integración, se abre un diálogo sobre los desafíos que enfrenta SOA y las oportunidades de investigación futura.

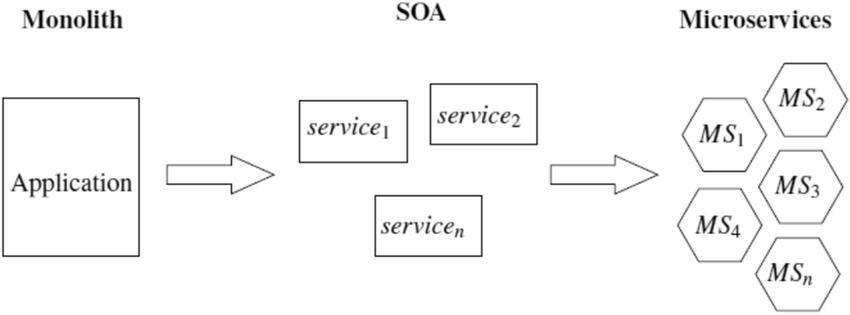
En resumen, SOA tiene el potencial de transformar la forma en que las organizaciones diseñan y gestionan sus aplicaciones, pero es crucial abordar los malentendidos que la rodean para maximizar su implementación efectiva y su impacto en el negocio.

Reflexión:

La Arquitectura Orientada a Servicios (SOA) emerge como un enfoque fundamental para la creación de aplicaciones ágiles y flexibles en un entorno empresarial en constante cambio. Su diseño jerárquico de servicios no solo permite la implementación simultánea de procesos, sino que también promueve la adaptabilidad, un aspecto crucial en la actualidad. Sin embargo, la adopción de SOA ha enfrentado obstáculos debido a malentendidos que han afectado su percepción y aplicación.

El artículo analiza no solo la historia y las bases tecnológicas de SOA, sino también su integración con plataformas modernas, destacando la relevancia de la reutilización y el acoplamiento flexible. Al investigar metodologías de diseño y aplicaciones de integración, se plantea un espacio para discutir los desafíos actuales y las futuras direcciones de investigación.

En conclusión, aunque SOA tiene el potencial de revolucionar la gestión y diseño de aplicaciones en las organizaciones, es fundamental desmitificar sus conceptos y beneficios para asegurar una implementación exitosa y maximizar su impacto positivo en el negoci

****

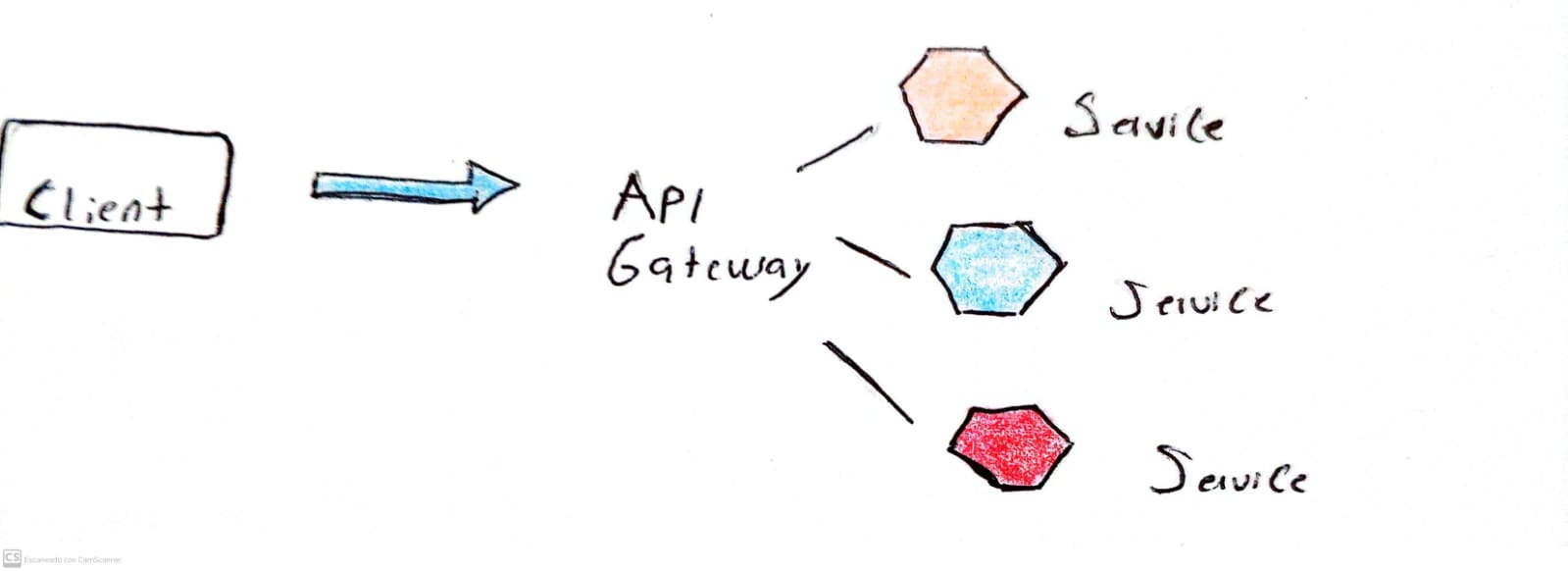
**Monolitos vs. Microservicio en Arquitectura de Software: Perspectivas para un Desarrollo Eficiente**

Este estudio analiza exhaustivamente dos enfoques clave en el desarrollo de software: los monolitos y los microservicios. Examina sus estructuras, interacciones de componentes, desafíos, ventajas y su implementación práctica en entornos reales. También evalúa casos de estudio de empresas líderes que han adoptado o migrado entre ambas arquitecturas. El análisis incluye el despliegue, mantenimiento y escalabilidad a largo plazo, proporcionando una visión completa para que los profesionales puedan tomar decisiones informadas sobre qué arquitectura utilizar, considerando las implicaciones técnicas, estratégicas y de costos a largo plazo.

Reflexión:

Este estudio ofrece un análisis profundo de dos enfoques en el desarrollo de software: monolitos y microservicios. Al examinar sus estructuras, ventajas y desafíos, se proporciona un contexto práctico mediante casos de estudio de empresas que han migrado entre ambas arquitecturas.

El enfoque en despliegue, mantenimiento y escalabilidad permite a los profesionales evaluar las implicaciones técnicas, estratégicas y de costos a largo plazo. En resumen, este análisis es esencial para que los desarrolladores tomen decisiones informadas y elijan la arquitectura más adecuada según las necesidades de sus proyectos.



**Bibliografías**

**1) resumen**

**Milić, M., & Makajić-Nikolić, D.** (2022). Development of a quality-based model for software architecture optimization: A case study of monolith and microservice architectures. Symmetry, 14(9), 1824. <https://doi.org/10.3390/sym14091824>

**2) resumen**

**McLarty, M.** (2016, May 30). Microservice architecture is agile software architecture. InfoWorld.com. Foundry. <https://login.bdigital.sena.edu.co/login?url=https://www.proquest.com/trade-journals/microservice-architecture-is-agile-software/docview/1792533407/se-2?accountid=31491>

**3) resumen**

**Saman, B.** (2017). Monitoring and analysis of microservices performance. Journal of Computer Science and Control Systems, 10(1), 19–22. <https://login.bdigital.sena.edu.co/login?url=https://www.proquest.com/scholarly-journals/monitoring-analysis-microservices-performance/docview/1937671608/se-2?accountid=31491>

**4) resumen**

**Tapia, F., Mora, M. Á., Fuertes, W., Aules, H., Flores, E., & Toulkeridis, T.** (2020). From monolithic systems to microservices: A comparative study of performance. Applied Sciences, 10(17), 5797. <https://doi.org/10.3390/app10175797>

**5) resumen**

**Strîmbei, C., Dospinescu, O., Strainu, R.-M., & Nistor, A.** (2015). Software architectures - Present and visions. Informatica Economica, 19(4), 13–27. <https://doi.org/10.12948/issn14531305/19.4.2015.02>

**6) resumen**

**Milić, M., & Makajić-Nikolić, D.** (2022). Development of a quality-based model for software architecture optimization: A case study of monolith and microservice architectures. Symmetry, 14(9), 1824. <https://doi.org/10.3390/sym14091824>

**7) resumen**

**Fruhlinger, J.** (2019, October 10). What are microservices? Your next software architecture. InfoWorld.com. Foundry. <https://login.bdigital.sena.edu.co/login?url=https://www.proquest.com/trade-journals/what-are-microservices-your-next-software/docview/2303450345/se-2?accountid=31491>

**8) resumen**

**Barnes, J. M., Garlan, D., & Schmerl, B.** (2014). Evolution styles: Foundations and models for software architecture evolution. Software and Systems Modeling, 13(2), 649–678. <https://doi.org/10.1007/s10270-012-0301-9>

**9) resumen**

**Lambert, N.** (2016, November 22). Microservices: Breaking down software monoliths. Network World (Online). Foundry. <https://login.bdigital.sena.edu.co/login?url=https://www.proquest.com/trade-journals/micro-services-breaking-down-software-monoliths/docview/1843839075/se-2?accountid=31491>

**10) resumen**

**Ji, L., Singh, A., Sharma, A. K., Joshi, A., Yadav, N. S., & Verma, A.** (2022). A comprehensive analysis of the literature on the "traditional" monolithic approach and microservices architecture. NeuroQuantology, 20(15), 7695-7699. <https://doi.org/10.14704/NQ.2022.20.15.NQ88760>

**11) resumen**

**Kaur, P., Singhal, H., Saxena, A., Mittal, N., & Dabas, C.** (2021). Polyglot persistence for microservices-based applications. International Journal of Information Technologies and Systems Approach, 14(1), 17-32. <https://doi.org/10.4018/IJITSA.2021010102>

**12) resumen**

**Santos, S., & Silva, A. R.** (2022). Microservices identification in monolith systems: Functionality redesign complexity and evaluation of similarity measures. Journal of Web Engineering, 21(5), 1543-1582. <https://doi.org/10.13052/jwe1540-9589.2158>

**13) resumen**

**Santos, S., & Silva, A. R.** (2022). Microservices identification in monolith systems: Functionality redesign complexity and evaluation of similarity measures. Journal of Web Engineering, 21(5), 1543-1582. <https://doi.org/10.13052/jwe1540-9589.2158>

**14) resumen**

**Mathai, A., Bandyopadhyay, S., Desai, U., & Tamilselvam, S.** (2022). Monolith to microservices: Representing application software through heterogeneous graph neural network. arXiv.org. <http://arxiv.org/abs/2112.01317>

**15) resumen**

**Al-Debagy, O., & Martinek, P.** (2019). A new decomposition method for designing microservices. Periodica Polytechnica. Electrical Engineering and Computer Science, 63(4), 274-281. <https://doi.org/10.3311/PPee.13925>

**16) resumen**

**Mamani Rodríguez, Z., Del Pino Rodríguez, L., & Gonzales Suarez, J. C.** (2020). Arquitectura basada en microservicios y DevOps para una ingeniería de software continua. Industrial Data, 23(2). <https://doi.org/10.15381/idata.v23i2.17278>

**17) resumen**

**Raj, V., & Bhukya, H.** (2023). Assessing the impact of migration from SOA to microservices architecture. SN Computer Science, 4(5), 577. <https://doi.org/10.1007/s42979-023-01971-2>

**18) resumen**

**Weerasinghe, S., & Perera, I.** (2023). Optimized strategy for inter-service communication in microservices. International Journal of Advanced Computer Science and Applications, 14(2). <https://doi.org/10.14569/IJACSA.2023.0140233>

**19) resumen**

Ren, M., & Lyytinen, K. J. (2008). Building enterprise architecture agility and sustenance with SOA. Communications of the Association for Information Systems, 22, 4. <https://doi.org/10.17705/1CAIS.02204>

**20) resumen**

Torassa Colombero, V., Estelles, J., Gallegos, L., & Lopez, P. (2024). Monolitos vs. Microservicios en arquitectura de software: Perspectivas para un desarrollo eficiente. Memorias de las JAIIO, 10(5), 42-54. <https://publicaciones.sadio.org.ar/index.php/JAIIO/article/view/928>