REVISTA TIA

Tecnología, Investigación y Academia -Red Avanzada – RITA
Publicación Facultad de Ingeniería y Red de Investigaciones de Tecnología

Análisis de un sistema de registro unificado de eventos en una plataforma con una arquitectura microservicios para la toma acertada de decisiones

Analysis of a unified event logging system on a platform with a microservices architecture for accurate decision making.

Juan Sebastián Calderón Ríos[1], Diego Fernando Melo Vaquen[2]

Citar este documento:

Calderón Ríos - Juan Sebastián, Melo Vaquen- Diego Fernando 2023. Análisis de un sistema de registro unificado de eventos en una plataforma con una arquitectura microservicios para la toma acertada de decisiones, Revista Tecnol.Investig. Academia TIA, ISSN: 2344-8288, Vol. 11 No 1, pp. 89-101. Bogotá-Colombia.

[1] Ingeniero de desarrollo, U.D. Francisco José de Caldas, Aldeamo, jscalderonr@correo.udistrital.edu.co, Colombia https://orcid.org/0000-0002-9570-8321

[2] Ingeniero de. Telecomunicaciones, Politecnico Grancolombiano , SENA, dfmelov@correo.udistrital.edu.co, Colombia https://orcid.org/0000-0001-5622-1804

Artículo corto
Fecha de recepción: noviembre -2022.
Fecha de aceptación: enero 2023
ISSN: 2344 8288
Vol. 11 No. 1

Resumen:

En la actualidad el avance tecnológico nos ha permitido implementar nuevos métodos para la toma rápida y acertada de decisiones dentro de las organizaciones basándose en información confiable y segura. La mayoría de los sistemas de información gerenciales guardan información conocida como rastros de auditoría que ayudan a determinar el tiempo y/o las personas que hayan realizado o modificado la información registrada, debido a esto se evidencia la importancia de implementar y gestionar un registro unificado de eventos dentro en las organizaciones que permita determinar riesgos en diferentes puntos del manejo de la información, para garantizar la calidad, disponibilidad y pertinencia de dicha información. [16] Este artículo estudia los métodos que ofrece el mercado actual para el análisis y evaluación de una serie de eventos filtrando eventos de error que provienen de una plataforma compuesta por N microservicios con flujos independientes con el fin de reducir el trabajo humano y optimizar el proceso.

Palabras clave*: aplicaciones web, gestión de la información, microservicios, software*

Abstract:

At present, technological advancement has allowed us to implement new methods for quick and accurate decision-making within organizations based on reliable and secure information. Most management information systems save information known as audit trails that help determine the time and/or people who have made or modified the recorded information, due to this the importance of implementing and managing a unified record of events within organizations that allow determining risks at different points of information management, to guarantee the quality, availability, and relevance of this information. This article studies the methods offered by the current market for the analysis and evaluation of a number of events filtering error events that come from a platform composed of N microservices with independent flows in order to reduce human work and optimize the process.

Keywords: *web applications, information management, microservices, software*

I. INTRODUCCIÓN

La toma de decisiones en las organizaciones bien sea para su gestión administrativa, operativa o relacionada con el núcleo (Core) del negocio, tienen implicaciones que pueden representar mejoras en su desempeño y productividad o por el contrario riesgos de pérdidas operativas o productivas [1]. Generalmente, estas decisiones se toman basadas en la información contenida en sus estructuras de datos y con muy poca frecuencia se aplican mecanismos de validación de la calidad de dichos datos, lo que puede representar riesgos no atendidos [2, 3]

El personal que frecuentemente hace uso de los datos, la frecuencia de intervención sobre los mismos, la rotación de personal que tiene acceso a la información y los riesgos de afectación sobre los datos, bien sean de manera accidental o intencional, son factores constantes que pueden afectar la calidad de los datos, sobre los que se toman decisiones en las organizaciones.

La toma de decisiones tiene un carácter informacional por la marcada dependencia a la información como recurso estratégico. En el máximo nivel de decisión organizacional se requieren no solo programas y procedimientos que reduzcan la incertidumbre, sino competencias, mecanismos, dinámicas y capacidades organizacionales que permitan a los decisores tomar acertadas decisiones estratégicas.[15]

La construcción de un registro unificado de eventos en las organizaciones permitirá identificar algún tipo de falencia en algún punto de una plataforma basada en una arquitectura de microservicios ya sea durante la entrada, proceso o salida de esta, dado que ayudará a determinar quién, cuándo, dónde, por qué y a través de qué medio se ha realizado la utilización y/o modificaciones de la información ya sea por los diferentes usuarios o procesos que intervienen en la misma.

II. ANÁLISIS

Los microservicios, en un estilo de arquitectura una aplicación está compuesta por componentes discretos, conectados en red, denominados microservicios. El estilo arquitectónico de microservicios es una evolución del estilo arquitectónico SOA (Arquitectura orientada a los

Artículo corto ISSN: 2344-8288 Vol. 11 No. 1. 2023 Bogotá-Colombia

servicios). Las aplicaciones construidas mediante los servicios SOA tienden a enfocarse en problemas de integración técnica, y el nivel de los servicios implementados es a menudo el de las interfaces de programación de aplicaciones (API) técnicas de grano fino. En contraste, el enfoque de microservicios implementa claras capacidades empresariales a través de API empresariales de grano grueso. [4]

Arquitectura de microservicios

Las arquitecturas de microservicios son un enfoque para desarrollar aplicaciones de servidor como un grupo de pequeños servicios. Cada servicio se ejecuta en su propio proceso y se comunica con otros procesos usando distintos protocolos, como HTTP/HTTPS o WebSockets. Cada microservicio es independiente de los demás y por tanto implementa una funcionalidad o lógica de negocio completa, de principio a fin. Debe ser desarrollado autónomamente y desplegado independientemente. Además, la lógica del dominio implementada en el microservicio será responsable de sus propios modelos de almacenamiento de datos, es decir, esta arquitectura contará con un grupo de servicios los cuales serán autónomos e independientes de cada uno de ellos, implementando una funcionalidad de negocio individual que permite tener un contexto ilimitado de servicios dentro de una organización. [5,6]

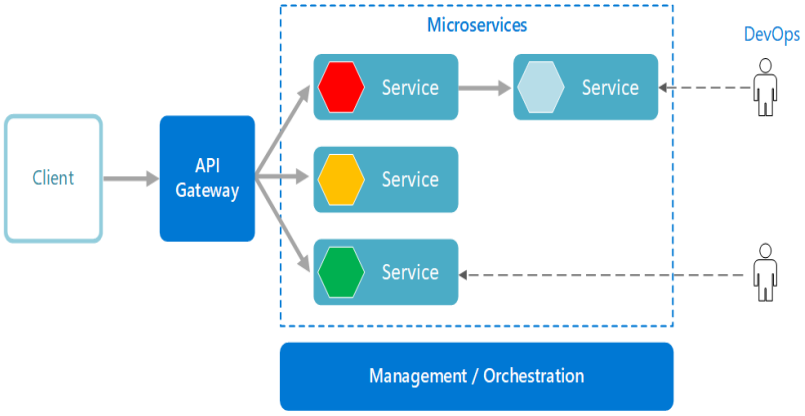

Figura 1. Estilo de arquitectura de microservicios

Fuente [6]

Las aplicaciones basadas en arquitecturas de microservicios facilitan escenarios de integración y entrega continuas, elementos que son parte de la cultura DevOps. Aceleran la entrega de nuevas

funcionalidades en la aplicación y permiten su evolución de forma autónoma siempre que los contratos que éstas exponen se mantengan claros. Si no los cambiamos podremos modificar la implementación de cualquier microservicio o añadir nuevas funcionalidades sin afectar a otros microservicios. [7]

En comparación con los aplicativos desarrollados en arquitectura monolítica tradicional (en esta únicamente se emplea una tecnología en desarrollo) los cuales limitan la disponibilidad de la información, debido a que presenta, procesa y almacena la información dentro de un mismo componente de software, siendo ejecutado en un único servidor [8], la arquitectura de microservicios facilita la ejecución de diferentes procesos al mismo tiempo sin afectar el rendimiento para la obtención de la información.

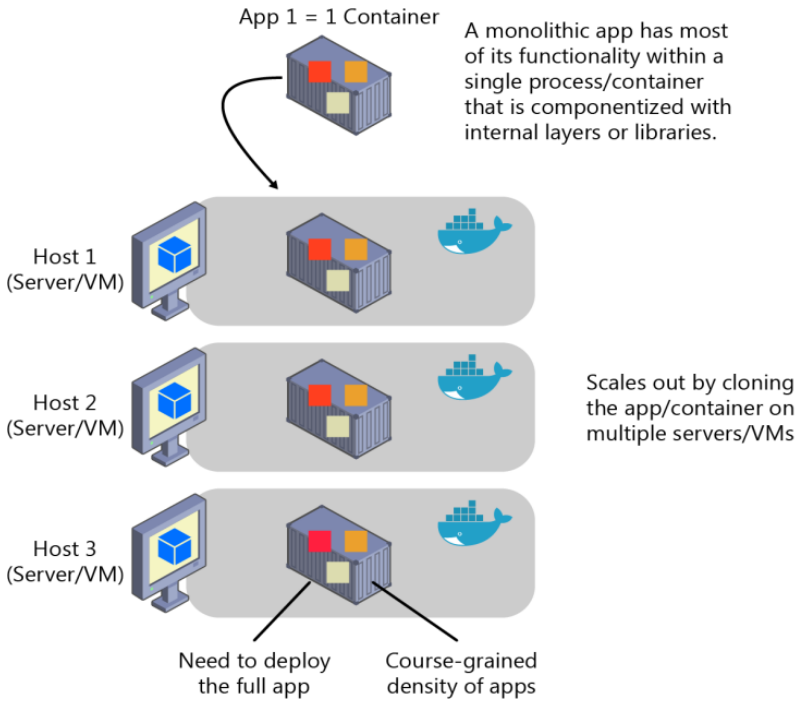

Figura 2. Estilo de arquitectura monolítica Fuente [9]

Veamos qué ventajas y desventajas [10] tiene la aplicación de una arquitectura de microservicios frente a otros tipos de arquitectura:

Ventajas:

- Modularidad: al tratarse de servicios autónomos, se pueden desarrollar y desplegar de forma independiente. Además, un error en un servicio no debería afectar la capacidad de otros servicios para seguir trabajando según lo previsto.
- Escalabilidad: como es una aplicación modular, se puede escalar horizontalmente cada parte según sea necesario, aumentando el escalado de los módulos que tengan un procesamiento más intensivo.
- Versatilidad: se pueden usar diferentes tecnologías y lenguajes de programación. Lo que permite adaptar cada funcionalidad a la tecnología más adecuada y rentable.
- Rapidez de actuación: el reducido tamaño de los microservicios permite un desarrollo menos costoso, así como el uso de "contenedores de software" permite que el despliegue de la aplicación se pueda llevar a cabo rápidamente.
- Mantenimiento simple y barato: al poder hacerse mejoras de un solo módulo y no tener que intervenir en toda la estructura, el mantenimiento es más sencillo y barato que en otras arquitecturas.
- Agilidad: se pueden utilizar funcionalidades típicas (autenticación, trazabilidad, etc.) que ya han sido desarrolladas por terceros, no hace falta que el desarrollador las cree de nuevo.

Desventajas:

- Alto consumo de memoria: al tener cada microservicio sus propios recursos y bases de datos, consumen más memoria y CPU.
- Inversión de tiempo inicial: al crear la arquitectura, se necesita más tiempo para poder fragmentar los distintos microservicios e implementar la comunicación entre ellos.
- Complejidad en la gestión: si contamos con un gran número de microservicios, será más complicado controlar la gestión e integración de los mismos. Es necesario disponer de una centralización de trazas y herramientas avanzadas de procesamiento de información que permitan tener una visión general de todos los microservicios y orquesten el sistema.
- Perfil de desarrollador: los microservicios requieren desarrolladores experimentados con un nivel muy alto de experiencia y un control exhaustivo de las versiones. Además de conocimiento sobre solución de problemas como latencia en la red o balanceo de cargas.

- No uniformidad: aunque disponer de un equipo tecnológico diferente para cada uno de los servicios tiene sus ventajas, si no se gestiona correctamente, conducirá a un diseño y arquitectura de aplicación poco uniforme.
- Dificultad en la realización de pruebas: debido a que los componentes de la aplicación están distribuidos, las pruebas y test globales son más complicados de realizar.
- Coste de implantación alto: una arquitectura de microservicios puede suponer un alto coste de implantación debido a costes de infraestructura y pruebas distribuidas. [10]

Resiliencia de los microservicios

En una arquitectura de microservicios se debe tener una alta resiliencia que podemos definir como la certeza en recuperarse ante un evento imprevisto y en este concepto interviene la alta disponibilidad (HA) y la recuperación de desastres (DR).

La HA asegura que los servicios estén disponibles para los usuarios finales cuando se realizan en el sistema actividades de mantenimiento como despliegue de actualizaciones, reinicio de las máquinas virtuales de “hosting”, y aplicación de parches de seguridad al SO de “hosting”. [10]

La HA generalmente no se aplica a problemas importantes inesperados, como la pérdida completa del sitio debido a grandes cortes de energía, terremotos, graves fallos de hardware o pérdida de la conectividad en todo el sitio. En tales casos, si los servicios tienen estrictos objetivos de nivel de servicio (SLO), se debería hacer redundante a toda la pila de aplicaciones (infraestructura, servicios y componentes de aplicaciones) recurriendo al menos a dos regiones diferentes. Esto se define típicamente como arquitectura de Recuperación de desastres (DR).

Registro unificado de eventos

En las aplicaciones complejas en algún momento algo puede salir mal. En una aplicación de microservicios se necesita realizar un seguimiento de lo que está sucediendo en docenas o incluso centenares de servicios. Para comprender lo que está pasando, se debe recopilar registros y métricas.

Los registros son registros de eventos basados en texto que tienen lugar mientras la aplicación está en ejecución. Incluyen elementos como registros de aplicación (instrucciones de seguimiento) o registros de servidor web. Los registros son sobre todo útiles para analizar la causa raíz y los análisis forenses. Estos son algunos de los desafíos generales del registro en una aplicación de microservicios [11]:

- Comprender el procesamiento de un extremo a otro de una solicitud de cliente, en la que se pueden invocar varios servicios para controlar una única solicitud.
- Consolidar los registros de varios servicios en una única vista agregada.
- Analizar los registros que proceden de varios orígenes, que usan sus propios esquemas de registro o que no tienen un esquema determinado. Los registros pueden haber sido generados por componentes de terceros sobre los que no tiene control.
- Las arquitecturas de microservicios suelen generar un volumen mayor de registros que las arquitecturas monolíticas tradicionales, ya que hay más servicios, llamadas de red y pasos en una transacción. Esto significa que el propio registro puede ser un cuello de botella de rendimiento o de recursos para la aplicación.

Las métricas son valores numéricos que se pueden analizar, estas pueden ser usadas para observar el sistema en tiempo real (o casi en tiempo real) o para analizar las tendencias de rendimiento a lo largo del tiempo. Para comprender el sistema de forma holística, se deben recopilar métricas en distintos niveles de la arquitectura, desde la infraestructura física hasta la aplicación.

El Flujo de eventos en microservicios, por definición, un evento tiene la intención de informar sobre un hecho relevante que haya ocurridos y en el que un tercero podría estar interesado para determinar cómo podría ser otro servicio.

Un aspecto fundamental es la colaboración en un flujo de eventos, es decir, que todos los microservicios publicarán eventos cuando ocurra algo relevante para el negocio dentro de ellos. Otros servicios pueden suscribirse a este evento y hacer algo con él, almacenar la información

asociada en una forma óptima para sus propios fines. En algún momento posterior, un microservicio suscrito puede usar esa información para llevar a cabo su propio servicio sin depender de llamar a otros servicios. Por lo tanto, con la colaboración de eventos, un alto grado de desacoplamiento temporal entre servicios se convierte en un valor predeterminado. [12]

Esta colaboración de eventos podría implementarse con mensajería asincrónica, pero también podría implementarse por otros medios. Los microservicios podrían, por ejemplo, publicar feeds basados en REST de sus eventos que podrían ser consumidos por otros servicios de forma regular.

Microsoft propone un registro unificado que utiliza eventos discretos que le permitirá hacer un seguimiento y que estará en la capacidad de informar de los datos que generen las aplicaciones de forma centralizada. Cada registro proporciona información general sobre el estado de la ejecución de una aplicación, realizará seguimiento a códigos de error y entregarán mensajes informativos, en esta propuesta se plantea un umbral de error o un criterio de error. [13]

En esta propuesta se requiere de un flujo continuo de una aplicación, es decir se debe conocer de antemano el inicio y el fin de la aplicación, también se debe conocer el estado y las transiciones de los datos. Siempre que se produce un error en una aplicación, los equipos necesitan saber lo siguiente:

- ¿Por qué se ha producido el error en la aplicación?
- ¿Cuándo se ha producido la excepción de la aplicación?
- ¿Qué método ha provocado la excepción?
- ¿Qué eventos se registraron hasta el momento del error de la aplicación?
- ¿La excepción produjo posibles daños en los datos?

El registro, el seguimiento y la supervisión pueden proporcionar las respuestas a estas preguntas, así como la supervisión del uso y el rendimiento de la aplicación.

Para aplicaciones monolíticas el registro el seguimiento y la supervisión se producen en un dominio de un solo proceso, para desarrollo en la nube con una arquitectura en microservicios el registro y seguimiento resulta un poco más complejo dado que una única solicitud de aplicación

puede interactuar con muchos microservicios, cada microservicio genera su propio registro y determinar el flujo del proceso de ejecución de la aplicación puede ser difícil, para solucionar esto se plantea la siguiente arquitectura Figura 3, se usan los servicios de Azure para compilar un sistema de registro y supervisión unificados.

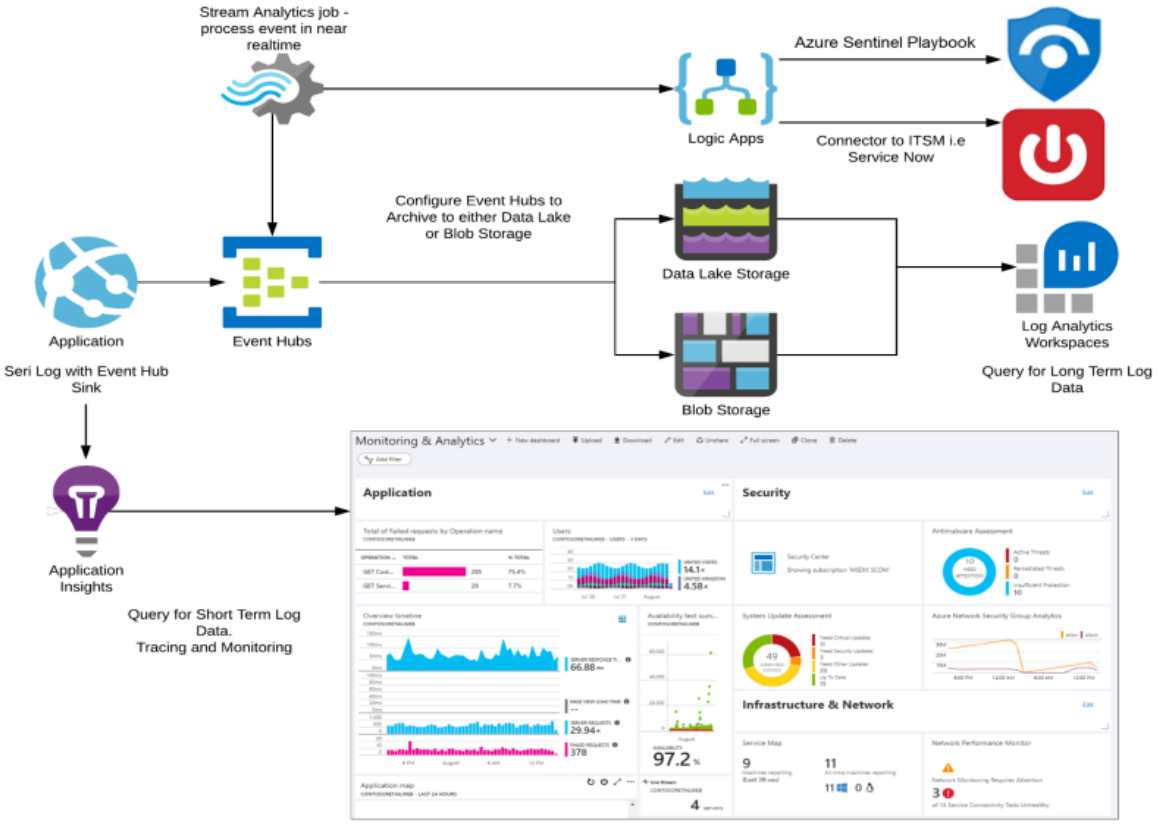

Figura 3. Arquitectura Azure para compilar un sistema de registro y supervisión unificados.

Fuente [13]

1. La aplicación emite eventos desde la API y la interfaz de usuario para Event Hubs y Application Insights.
2. Application Insights consulta datos de registro, seguimiento y supervisión a corto plazo.
3. Los trabajos de Stream Analytics pueden usar los datos de Event Hubs para desencadenar flujos de trabajo de Logic Apps.
4. Un trabajo de Logic Apps llama al punto de conexión de transferencia de estado representacional (REST) de un sistema de administración de servicios de TI (ITSM) y envía notificaciones al equipo de desarrollo.

5. La automatización de Azure Sentinel usa cuadernos de estrategias de Azure Logic Apps para generar alertas de seguridad.
6. El mantenimiento de registros de eventos en almacenamiento a largo plazo permite realizar análisis y diagnósticos posteriores con Log Analytics.

En la anterior arquitectura se observa que para la recopilación, análisis y posterior visualización de las trazas capturadas es necesario que varios componentes interactúen de manera coordinada entre si buscando que la información sea coherente para que lo presentado sea acertado. De manera que se requiere que la misma arquitectura sea en un principio bien configurada y por el tiempo en que se use mantenida de manera supervisada.

Niveles de gravedad del registro de eventos y seguimiento

En la definición de eventos o registros de seguimiento se deben determinar los valores que indican la gravedad de estos con el objetivo de limitar peticiones repetidas que pudieran desbordar los archivos de registro. Para lo anterior se plantea los siguientes niveles de gravedad para eventos Tabla 1:

Nombre de nivel	Id. de nivel	Se muestra en el registro de eventos como	Descripción
Error crítico	30	Crítico	Eventos que requieren la atención inmediata del administrador del sistema. Por lo general, se dirigen al nivel global (todo el sistema), como, por ejemplo, el sistema o la aplicación. También se pueden usar para indicar que se produjo un error en una aplicación o sistema, o que uno de ellos dejó de responder.
Error	40	Error	Eventos que indican problemas, pero en una categoría que no requiere atención inmediata.
Advertencia	50	Advertencia	Eventos que proporcionan advertencias previas de posibles problemas; aunque no son una respuesta a un error real, una advertencia indica que un componente o aplicación no se encuentra en un estado ideal y que si se realizan más acciones podría producirse un error crítico.
Información	80	Información	Eventos que pasan información no crítica al administrador, similar a una nota que dice: "Informativo".
Detallado	100	Informativo	Estado detallado, como mensajes de progreso o de operación correcta.

Artículo corto ISSN: 2344-8288 Vol. 11 No. 1. 2023 Bogotá-Colombia

III. CONCLUSIONES

Para la toma acertada de decisiones en una organización es necesario contar con fuentes de información verídicas que permitan a los decisores plantear estrategias adecuadas ante problemas de estudio, el registro unificado de eventos se convierte así en una fuente de información capaz de analizar y filtrar la información relevante de gran cantidad de sucesos que se presenten en una plataforma y que de una u otra manera van a contribuir en este proceso de toma de decisiones. En una plataforma compuesta de muchos componentes individuales cada uno con funciones en específico, debe tener una visibilidad general de lo que está sucediendo en un intervalo de tiempo determinado y ser capaz de extraer información relevante, lo cual se convierte en una tarea compleja si no está automatizada; un mecanismo como el que se plantea en el estudio de este articulo sería la solución a esta necesidad.

REFERENCIAS

[1] Y. M. Abu Amuna, M. J. Al Shobaki and S. S. Abu Naser, "The Role of Knowledge-Based Computerized Management Information Systems in the Administrative Decision-Making Process," Int. j. inf. technol. electr. Eng., vol. 6, no. 2, pp. 1-9, April 2017. Disponible: http://dstore.alazhar.edu.ps/xmlui/handle/123456789/

[2] M. J. Al Shobaki and S. S. Abu Naser, "Decision support systems and its role in developing the universities strategic management: Islamic university in Gaza as a case study," Int. j. adv. res. dev., vol. 1, no. 10, pp. 33-47, October 2016. Disponible: https://core.ac.uk/download/pdf/131209194.pdf

[3] M. J. Al Shobaki and S. S. Abu Naser. "Performance development and its relationship to demographic variables among users of computerized management information systems in Gaza electricity Distribution Company," Int. j. humanit. soc. sci. res., vol. 2, no. 10, pp. 21-30, October 2016. Disponible: https://philarchive.org/archive/SHOPDAv1

[4] R. Barcia, K, Brown, R. Osowski. (2018, enero) Guía para punto de vista de microservicios. IBM. [online]. Disponible: https://www.ibm.com/downloads/cas/ODGVKQE7

[5] J. Lewis, M. Fowler. (2014, March). Microservices. [Online]. Disponible: https://martinfowler.com/articles/microservices.html

[6] Microsoft. (2019, Noviembre). Estilo de Arquitectura de microservicios. [Online]. Disponible: https://docs.microsoft.com/es-es/azure/architecture/guide/architecture-styles/microservices

[7] J. Ortega. (2020). Tecnologías para arquitecturas basadas en microservicios: Patrones y soluciones para aplicaciones desplegadas en contenedores. Valencia: Editorial académica española, 2020

[8] A.F. Saransig. C. "Análisis de rendimiento entre una arquitectura monolítica y una arquitectura de microservicios – tecnología basada en contenedores", Tesis maestría, Universidad Técnica del Norte. Ibarra. Ecuador. 2018

[9] Microsoft. Aplicaciones monolíticas. [Online]. Disponible: https://docs.microsoft.com/es-es/dotnet/architecture/containerized-lifecycle/design-develop-containerized-apps/monolithic-applications

[10] C. Albasanz. (2019, septiembre) Arquitectura de microservicios: qué es, ventajas y desventajas. [Online]. Disponible: https://decidesoluciones.es/arquitectura-de-microservicios/

[11] Microsoft (2021, abril). Supervisión de una arquitectura de microservicios en Azure Kubernetes Service (AKS). [Online]. Disponible: https://docs.microsoft.com/es-es/azure/architecture/microservices/logging-monitoring

[12] B. Rücker, M. Schimak (2017, June). Know the Flow! Microservices and Event Choreographies. [Online]. Disponible: https://www.infoq.com/articles/microservice-event-choreographies/

[13] Microsoft. Registro unificado para aplicaciones de microservicios. [Online]. Disponible: https://docs.microsoft.com/es-es/azure/architecture/example-scenario/logging/unified-logging

[14] Microsoft. Niveles de gravedad del registro de eventos y seguimiento. [Online]. Disponible: https://docs.microsoft.com/es-es/previous-versions/office/developer/sharepoint-2010/ff604025(v=office.14)

[15] Y. Rodríguez, M. Pinto (Abril 2018). Modelo de uso de información para la toma de decisiones estratégicas en organizaciones de información.

[16] Oscar Fruits, María Guevara, Tanya Magaly, Jennifer Avilés, Lorena Bravo (Abril 2018). Importancia de realizar auditoria de sistemas preventiva en las Organizaciones https://www.coursehero.com/file/66015214/246-740-1-PBpdf/

Publicación Facultad de Ingeniería y Red de Investigaciones de Tecnología Avanzada – RITA

REVISTA

TIA