EthAIca. 2024; 3:147
doi: 10.56294/ai2024147

REVISIÓN

Simplification of microservice development through low-code tools

Simplificación del desarrollo de microservicios mediante herramientas Low-code

Tomás Darquier[1], Pablo Alejandro Virgolini[1]

[1]Universidad Siglo 21, Licenciatura en Informática, S.C de Bariloche. Argentina.

Citar como: Darquier T, Virgolini PA. Simplification of microservice development through low-code tools. EthAIca. 2024; 3:147. https://doi.org/10.56294/ai2024147

Enviado: 03-09-2023 **Revisado:** 21-01-2024 **Aceptado:** 05-06-2024 **Publicado:** 06-06-2024

Editor: PhD. Rubén González Vallejo

ABSTRACT

Introduction: enterprise application development has undergone a significant transformation in recent years, driven by the need for scalable and flexible solutions. This evolution favored the adoption of microservice architectures, which, while offering benefits such as resilience and modular scalability, also introduced technical complexities in integration and maintenance. Against this backdrop, low-code tools emerged in response to the demand for more accessible and faster-to-implement solutions.
Development: the study addressed the technical fundamentals of microservice architectures, characterized by their ability to divide applications into independent services. However, these architectures posed challenges in terms of interoperability, the use of multiple languages, and increased testing and deployment efforts. Various commercial and academic platforms proposed low-code-based solutions aimed at simplifying microservice design through visual interfaces and reusable components. The technological proposal of the work integrated modern tools such as Java, Spring, Docker, and Kubernetes, along with visual development approaches, with the aim of facilitating the creation of efficient distributed systems.
Conclusions: the research concluded that low-code tools offered an effective way to mitigate the complexity inherent in microservices. These solutions reduced the technical burden without compromising software quality, promoting more inclusive and sustainable development environments. Likewise, the comparative analysis of platforms highlighted the need to continue refining these tools to achieve greater flexibility and standardization.

Keywords: Microservices; Low-code; Interoperability; Distributed architecture; Automation.

RESUMEN

Introducción: el desarrollo de aplicaciones empresariales experimentó una transformación significativa en los últimos años, motivado por la necesidad de soluciones escalables y flexibles. Esta evolución favoreció la adopción de arquitecturas de microservicios, las cuales, aunque ofrecieron beneficios como resiliencia y escalabilidad modular, también introdujeron complejidades técnicas en la integración y mantenimiento. Ante este panorama, surgieron herramientas Low-code como respuesta a la demanda de soluciones más accesibles y rápidas de implementar.
Desarrollo: el estudio abordó los fundamentos técnicos de las arquitecturas de microservicios, caracterizadas por su capacidad de dividir aplicaciones en servicios independientes. No obstante, estas arquitecturas implicaron desafíos en cuanto a interoperabilidad, uso de múltiples lenguajes y mayor esfuerzo en pruebas y despliegue. Diversas plataformas comerciales y académicas propusieron soluciones basadas en Low-code, orientadas a simplificar el diseño de microservicios a través de interfaces visuales y componentes reutilizables. La propuesta tecnológica del trabajo integró herramientas modernas como Java, Spring, Docker y Kubernetes, junto con enfoques visuales de desarrollo, con el objetivo de facilitar la creación de sistemas distribuidos eficientes.
Conclusiones: la investigación permitió concluir que las herramientas Low-code ofrecieron una vía efectiva

para mitigar la complejidad inherente a los microservicios. Estas soluciones permitieron reducir la carga técnica sin comprometer la calidad del software, promoviendo entornos de desarrollo más inclusivos y sostenibles. Asimismo, el análisis comparativo de plataformas destacó la necesidad de seguir perfeccionando estas herramientas para alcanzar mayor flexibilidad y estandarización.

Palabras clave: Microservicios; Low-code; Interoperabilidad; Arquitectura Distribuida; Automatización.

INTRODUCCIÓN

En los últimos años, el desarrollo de aplicaciones empresariales ha evolucionado significativamente, impulsado por la necesidad de escalabilidad, flexibilidad y tiempos de entrega más ágiles. Este contexto ha favorecido la adopción de arquitecturas basadas en microservicios, un enfoque que permite dividir una aplicación en componentes pequeños, independientes y fácilmente desplegables. Esta metodología, si bien ofrece beneficios como la resiliencia del sistema y la escalabilidad individualizada de componentes, también introduce desafíos técnicos importantes. Entre ellos, destacan la complejidad de integración, la gestión de múltiples servicios desarrollados en distintos lenguajes y la necesidad de herramientas especializadas para asegurar la interoperabilidad entre componentes.

Autores como Newman[1] y Fowler han resaltado estas dificultades, señalando cómo la implementación de microservicios requiere recursos técnicos avanzados, así como un esfuerzo adicional en las etapas de desarrollo, pruebas y mantenimiento. A pesar de estas barreras, la demanda por soluciones más ágiles ha motivado la aparición de plataformas que buscan abstraer dicha complejidad. En este escenario emergen las herramientas Low-code, cuya propuesta consiste en permitir el desarrollo de sistemas complejos mediante interfaces gráficas y elementos preconfigurados, reduciendo así la necesidad de programación manual.

Empresas como Microsoft (con Power Apps) o BettyBlocks han introducido plataformas orientadas a simplificar el diseño de arquitecturas distribuidas, democratizando el acceso al desarrollo de software incluso para usuarios con conocimientos técnicos limitados. Este enfoque ha sido también abordado por investigaciones académicas que proponen lenguajes específicos de dominio (DSL) y entornos visuales basados en técnicas como drag and drop o bloques gráficos, similares a los utilizados en plataformas educativas como Scratch. [2]

En este trabajo se plantea una propuesta tecnológica que busca optimizar el desarrollo de microservicios mediante herramientas Low-code, integrando tecnologías modernas como Java, Spring, PostgreSQL, Docker y Kubernetes, junto con interfaces visuales que facilitan la creación e integración de servicios. La investigación se sustenta en un marco teórico que analiza tanto los fundamentos técnicos de los microservicios como el impacto de las soluciones Low-code en la productividad del desarrollo. A través de esta iniciativa, se pretende contribuir al diseño de sistemas más accesibles, escalables y sostenibles, reduciendo la carga técnica sin sacrificar la calidad y robustez del software resultante.

DESARROLLO

MARCO TEÓRICO REFERENCIAL

Dominio del Problema

Inicialmente, se debe definir a que nos referimos al hablar de una arquitectura distribuida de microservicios. Como señala Newman[1], “las arquitecturas distribuidas de microservicios dividen una aplicación en servicios pequeños, autónomos y desplegables de forma independiente, lo que permite a las organizaciones escalar componentes específicos según sea necesario y mejorar la resiliencia general del sistema”.

Este tipo de arquitecturas permite la interoperabilidad entre microservicios desarrollados en diferentes lenguajes de programación, debido a sus estándares de comunicación como lo son REST o gRPC y sus contratos de servicio.

La interoperabilidad en arquitecturas de microservicios es crucial, dado que los servicios pueden estar construidos en diferentes lenguajes de programación o plataformas. Mantener estándares comunes en las interfaces y contratos de servicio es fundamental para garantizar que estos servicios puedan comunicarse de manera efectiva, aunque esto también introduce complejidades adicionales en el proceso de desarrollo y mantenimiento.

Las complejidades mencionadas, inherentes en las arquitecturas de microservicios, pueden dilatar los tiempos de desarrollo, pruebas e integración, afectando las capacidades productivas de una empresa o desarrollador.

Estos sistemas normalmente están diseñados para ser escalables y resilientes, pero eso no viene sin costo. Puede añadir complejidad al proceso de desarrollo, ya que es otro sistema que podrías necesitar ejecutar para desarrollar y probar tus servicios. También pueden ser necesarias máquinas adicionales y experiencia para mantener esta infraestructura en funcionamiento.[1]

Debido a lo expuesto en la cita, muchas empresas decidieron brindar al mercado herramientas para intentar

mitigar los conflictos expuestos. Es por esto por lo que, herramientas web como Power Apps de Microsoft o BettyBlocks, tuvieron su génesis brindando a los desarrolladores abstracciones de la excesiva complejidad presentada por las arquitecturas distribuidas, permitiendo el desarrollo de aplicaciones de esta índole de una forma más sencilla mediante iniciativas Low-code.

A su vez, la solución propuesta referida a la simplificación del desarrollo de microservicios mediante herramientas Low-code, fue, y sigue siendo investigado por diversos autores del ámbito académico, evidenciando la presencia de la problemática tratada. Como Chaudhary y Margaria[2] mediante su solución ideada basada en DSL y drag and drop, o también, la propuesta de Dhoke y Lokulwar, soportada en una interfaz similar a la presente en la plataforma Scratch, con el objeto de permitir la programación de servicios backend mediante bloques visuales preconstruidos y plantillas.

Tics

A continuación, se listan las tecnologías utilizadas en el desarrollo del proyecto:

Lenguajes de Programación

Java: "Java es un lenguaje multiplataforma, orientado a objetos y centrado en la red que se puede utilizar como una plataforma en sí mismo".

JavaScript: "JavaScript es un lenguaje de programación o de secuencias de comandos que te permite implementar funciones complejas en páginas web".[3]

Frameworks y Bibliotecas

Spring: "Spring Framework proporciona un modelo integral de programación y configuración para aplicaciones empresariales modernas basadas en Java, en cualquier tipo de plataforma de implementación".[4]

Apache Velocity: "Velocity es un motor de plantillas basado en Java. Permite a cualquiera utilizar un lenguaje de plantilla simple pero potente para hacer referencia a objetos definidos en código Java".[5]

Apache Jena: "Jena es un marco Java para crear aplicaciones de Web Semántica. Proporciona amplias bibliotecas de Java para ayudar a los desarrolladores a desarrollar código que maneje RDF, RDFS, RDFa, OWL y SPARQL".[5]

Thymeleaf: "El objetivo principal de Thymeleaf es aportar plantillas naturales y elegantes a tu flujo de trabajo de desarrollo HTML, que puede ser correctamente mostrado en los navegadores y también funcionar como prototipos estáticos.

Boostrap: "Bootstrap es un framework front-end utilizado para desarrollar aplicaciones web y sitios mobile first, o sea, con un layout que se adapta a la pantalla del dispositivo utilizado por el usuario".[6]

Motor de Base de Datos

PostgreSQL: "PostgreSQL, comúnmente pronunciado 'Post-GRES', es una base de datos de código abierto que tiene una sólida reputación por su fiabilidad, flexibilidad y soporte de estándares técnicos abiertos".[7]

Otras Tecnologías

GitHub: "GitHub es una plataforma donde puedes almacenar, compartir y trabajar junto con otros usuarios para escribir código".[8]

RDF: "RDF es un modelo estándar para el intercambio de datos en la Web. RDF tiene características que facilitan la combinación de datos incluso si los esquemas subyacentes difieren".

SPARQL: "SPARQL contiene capacidades para consultar patrones de gráficos requeridos y opcionales junto con sus conjunciones y disyunciones via RDF".

HTML5: "La última versión estable de HTML, HTML5 convierte a HTML de un simple formato de marcado para estructurar documentos en una plataforma completa de desarrollo de aplicaciones".[3]

CSS: "CSS, de las siglas en inglés Cascading Style Sheets (Hojas de Estilo en Cascada), es un lenguaje declarativo que controla el aspecto de las páginas web en el navegador".[3]

Docker: "Docker empaqueta software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución".

Kubernetes: "Kubernetes es una plataforma portable y extensible de código abierto para administrar cargas de trabajo y servicios. Kubernetes facilita la automatización y la configuración declarativa".[9]

MinIO: "Minio es un servidor de almacenamiento de objetos distribuidos de alto rendimiento, que está diseñado para la infraestructura de nube privada a gran escala".[7,10,11]

Apache Kafka: "Apache Kafka es una plataforma de transmisión de eventos distribuida de código abierto utilizada por miles de empresas para canalizaciones de datos de alto rendimiento, análisis de transmisión, integración de datos y aplicaciones críticas".[5,12]

IntelliJ IDEA: “IntelliJ IDEA incorpora uno de los editores de código más potentes del sector. Comprende los entresijos de su código gracias a la indexación inicial”.[13,14]

Postman: “Postman es una plataforma para crear y utilizar APIs. Postman simplifica cada paso del ciclo de vida y agiliza la colaboración para que puedas crear mejores APIs y más rápido”.[10,15,16]

Trello: “Trello es una herramienta visual que permite a los equipos gestionar cualquier tipo de proyecto y flujo de trabajo, así como supervisar tareas”.[17]

Competencia

A continuación, en la siguiente tabla se exponen y comparan las principales características de las distintas soluciones, similares a la propuesta en el presente proyecto, presentes en el mercado.

Tabla 1. Comparativa de empresas con soluciones Low-code

Páginas	Generación Automática de Microservicios	Soporte REST	Soporte gRPC	Integración con Sistemas Externos	Soporte para Contenedores
outsystems.com	X	X		X	X
mendix.com	X	X		X	X
appian.com	X	X		X	
microsoft.com/**/power-apps		X		X	
bettyblocks.com	X	X		X	X

CONCLUSIONES

A lo largo de este trabajo se ha evidenciado cómo la evolución en el desarrollo de aplicaciones empresariales ha generado nuevas demandas técnicas, particularmente en lo que respecta a la adopción de arquitecturas basadas en microservicios. Este enfoque, si bien ofrece beneficios sustanciales como la escalabilidad individualizada, resiliencia del sistema y una mayor flexi-bilidad en la implementación de componentes, también introduce una serie de desafíos complejos. La interoperabilidad entre servicios, la gestión de diferentes lenguajes de programación, la necesidad de infraestructura robusta y la sofisticación en el mantenimiento de sistemas distribuidos representan obstáculos importantes para los equipos de desarrollo.

Frente a esta complejidad técnica, las herramientas Low-code emergen como una solución estratégica que busca reducir la carga operativa sin comprometer la calidad del software. Al permitir el desarrollo de aplicaciones mediante interfaces visuales, elementos gráficos predefinidos y entornos intuitivos, estas plataformas democratizan el acceso al desarrollo de sistemas distribuidos, incluso para usuarios con conocimientos técnicos limitados. Las soluciones ofrecidas por empresas como Microsoft, BettyBlocks o plataformas académicas basadas en DSL y técnicas drag and drop, demuestran la viabilidad de esta tendencia y su impacto potencial en la productividad y eficiencia de los equipos de trabajo.

La propuesta presentada en este proyecto se inserta dentro de esta corriente innovadora, integrando tecnologías consolidadas como Java, Spring, PostgreSQL, Docker y Kubernetes, con herramientas Low-code que permiten construir, desplegar e integrar microservicios de forma más accesible. Esta combinación no solo simplifica el proceso técnico, sino que también promueve la creación de sistemas escalables y sostenibles, alineados con las exigencias del mercado actual.

El análisis comparativo realizado entre distintas plataformas Low-code permite destacar las fortalezas y limitaciones de las soluciones existentes, subrayando la necesidad de continuar desarrollando herramientas más completas, flexibles y orientadas a microservicios. A pesar de los avances logrados, persisten desafíos relacionados con la integración de estándares, la optimización del rendimiento y la capacitación de los usuarios en el uso eficaz de estas tecnologías.

Las herramientas Low-code representan una respuesta válida ante la complejidad de las arquitecturas modernas, ofreciendo un puente entre la sofisticación técnica y la necesidad de agilidad. La solución planteada en este trabajo aporta un enfoque equilibrado que puede contribuir significativamente al desarrollo de software más eficiente, accesible y adaptable a los constantes cambios del entorno tecnológico.

REFERENCIAS BIBLIOGRÁFICAS

1. Newman S. Building microservices: Designing fine-grained systems. Newton: O’Reilly Media; 2015.

2. Chaudhary HAA, Ahmed T. Integration of micro-services as components in modeling environments for low code development. ISP RAS. 2021;33(4):19-30. doi:10.15514/ISPRAS-2021-33(4)-2

3. Mozilla Developer Network. CSS. 2023. Available from: https://developer.mozilla.org/es/docs/Glossary/CSS

4. Spring. Spring Framework. 2024. Available from: https://spring.io/projects/spring-framework

5. Apache Software Foundation. Apache Kafka. 2024. Available from: https://kafka.apache.org/

6. Rock Content. What is Bootstrap? 2020. Available from: https://rockcontent.com/es/blog/bootstrap/

7. IBM. Minio. 2021. Available from: https://www.ibm.com/docs/es/cloud-private/3.2.x?topic=private-minio

8. GitHub. About GitHub and Git. 2024. Available from: https://docs.github.com/es/get-started/start-your-journey/about-github-and-git

9. DigitalOcean. DigitalOcean Managed Kubernetes. 2024. Available from: https://www.digitalocean.com/products/kubernetes

10. Postman. What is Postman? 2024. Available from: https://www.postman.com/product/what-is-postman/

11. Banco Central de la República Argentina. Cotizaciones por fecha. 2024. Available from: http://www.bcra.gob.ar/PublicacionesEstadisticas/Cotizaciones_por_fecha_2.asp

12. Brown T, Smith L. The impact of Low-code platforms and intuitive interfaces on software development efficiency. Journal of Software Innovation. 2023;18:75-89.

13. Consejo Profesional de Ciencias Informáticas de la Provincia de Córdoba. Honorarios Recomendados. 2024. Available from: https://cpcipc.org.ar/honorarios-recomendados/

14. Elgheriani NS, Ahmed NA. Microservices vs. monolithic architectures. International Journal of Applied Science and Technology. 2022;4:501-14. doi:10.47832/2717-8234.12.47

15. Misic B, Novkovic M, Ramac R, Mandic V. Do the microservices improve the agility of software development teams? In: International Scientific Conference on Industrial Systems; 2017. p. 170-5.

16. Schwaber K, Sutherland J. Scrum: The art of doing twice the work in half the time. Houston: Crown Business; 2010.

17. Trello. Trello Tour. 2023. Available from: https://trello.com/es/tour

FINANCIACIÓN

Ninguna.

CONFLICTO DE INTERESES

Los autores declaran que no existe conflicto de intereses.

CONTRIBUCIÓN DE AUTORÍA

Conceptualización: Tomás Darquier, Pablo Alejandro Virgolini.
Curación de datos: Tomás Darquier, Pablo Alejandro Virgolini.
Análisis formal: Tomás Darquier, Pablo Alejandro Virgolini.
Investigación: Tomás Darquier, Pablo Alejandro Virgolini.
Metodología: Tomás Darquier, Pablo Alejandro Virgolini.
Administración del proyecto: Tomás Darquier, Pablo Alejandro Virgolini.
Recursos: Tomás Darquier, Pablo Alejandro Virgolini.
Software: Tomás Darquier, Pablo Alejandro Virgolini.
Supervisión: Tomás Darquier, Pablo Alejandro Virgolini.
Validación: Tomás Darquier, Pablo Alejandro Virgolini.
Visualización: Tomás Darquier, Pablo Alejandro Virgolini.

Redacción – borrador original: Tomás Darquier, Pablo Alejandro Virgolini.
Redacción – revisión y edición: Tomás Darquier, Pablo Alejandro Virgolini.

https://doi.org/10.56294/ai2024147 ISSN: 3072-7952