© FEGLININ
| Año IO, No 39, octubre - diciembre 2026 |
https://doi.org/10.70417/_23AI0atIIj28
ISSN 2594-2298
Prefijo DOI: 10.70417

DE LOS PLCs A LA WEB: UNA ARQUITECTURA DE REFERENCIA PARA SISTEMAS ADAPTATIVOS DE GESTIÓN DE FLUJOS DE TRABAJO



FROM PLCs TO THE WEB: A REFERENCE ARCHITECTURE FOR ADAPTIVE WORKFLOW MANAGEMENT SYSTEMS

Ing. Mauricio Lujano Dávalos[1]

RESUMEN

Los Sistemas Adaptativos de Gestión de Flujos de Trabajo basados en web (Web-WFMS) carecen de bases arquitectónicas consolidadas. Se evalúa un modelo arquitectónico inspirado en PLCs para entornos web, proponiendo una arquitectura de referencia instanciada mediante ingeniería de prompts. La validación a través de un sistema de control IoT basado en reglas demuestra la viabilidad técnica de la solución. El criterio de éxito es una arquitectura de referencia que incorpora los componentes esenciales de un WFMS y se alinea con los impulsores arquitectónicos del proyecto. El análisis de literatura revela que los paradigmas reactivos ofrecen ganancias de rendimiento bajo alta concurrencia (Li & Guo, 2024; Luo et al., 2024; Zbarcea & Tudose, 2024), pero introducen entre un 15% y un 20% más de código y complejidad de depuración (Kumar, 2025; Iwanowski & Koziel, 2022), haciéndolos inadecuados para escenarios de baja carga (Kasianchuk, 2024). Las contribuciones incluyen: (a) caracterización de aspectos intrínsecos de WFMS, (b) una arquitectura de referencia con instanciación mediante ingeniería de prompts, (c) orientación fundamentada para seleccionar entornos tradicionales vs. reactivos, y (d) identificación de tecnologías WFMS establecidas (n8n, AWS Step Functions, Camunda) como referencias para consideración inmediata.

ABSTRACT

Adaptive Web-based Workflow Management Systems (Web-WFMS) lack consolidated architectural foundations. This study evaluates a PLC-inspired architectural model for web environments, proposing a reference architecture instantiated via prompt engineering. Validation through a rule-based IoT control system demonstrates the solution's technical feasibility. The success criterion is a reference architecture that incorporates the essential architectural components of a WFMS and aligns with the project's architectural drivers. Literature analysis reveals that reactive paradigms deliver performance gains under high concurrency (Li & Guo, 2024; Luo et al., 2024; Zbarcea & Tudose, 2024), but introduce 15–20% more code and debugging complexity (Kumar, 2025; Iwanowski & Koziel, 2022), making them unsuitable for low-load scenarios (Kasianchuk, 2024). Contributions include: (a) characterization of WFMS intrinsic aspects, (b) a reference architecture with prompt engineering instantiation, (c) evidence-based guidance for selecting traditional vs. reactive environments, and (d) identification of established WFMS technologies (n8n, AWS Step Functions, Camunda) as references for immediate consideration.

Palabras clave: Arquitectura de Software, Arquitecturas de Referencia, Sistemas de Gestión de Flujos de Trabajo, Programación Reactiva, Ingeniería de Prompts.

***Key words**: Software Architecture, Reference Architectures, Workflow Management Systems, Reactive Programming, Prompt Engineering.*

INTRODUCCIÓN

Los Sistemas Adaptativos de Gestión de Flujos de Trabajo basados en web (Web-WFMS) son cada vez más fundamentales para la automatización empresarial moderna, requiriendo

[1] Ing. Mauricio Lujano Dávalos, Instituto Tecnológico de Zitácuaro, M24650515@zitacuaro.tecnm.mx

monitoreo en tiempo real, orquestación basada en eventos y configuración dinámica basada en reglas. La evolución de los WFMS se originó en la necesidad de coordinación automatizada de recursos, con la Workflow Management Coalition (WfMC) estableciendo estándares fundamentales que siguen siendo influyentes (Represa et al., 2023; Hollingsworth, 1995).

Los sistemas de control industrial han aprovechado durante mucho tiempo los Controladores Lógicos Programables (PLC) para requisitos similares, demostrando la viabilidad de las arquitecturas basadas en eventos en dominios críticos (Ferencz et al., 2023; Vyatkin, 2011). Mohamed et al. (2023) describen los PLC como unidades de cómputo flexibles que monitorean continuamente entradas para controlar salidas, lo que establece una base para la analogía funcional entre PLC y WFMS. Sin embargo, la convergencia de estos dominios permanece poco explorada.

Los profesionales que desarrollan Web-WFMS adaptativos enfrentan tres problemas interrelacionados:

a**)** Falta de orientación arquitectónica sistemática**:** A pesar de la proliferación de arquitecturas WFMS—incluyendo núcleos RESTful (Bailón-Delgado et al., 2024), orquestación nativa en la nube (Olteanu et al., 2025) y microservicios (Negru et al., 2022)—los profesionales carecen de un marco consolidado que guíe sistemáticamente la selección de tecnologías y las decisiones de diseño arquitectónico (Bass et al., 2012; Dasanayake et al., 2015). Damarowsky et al. (2023) definen los WFMS modernos como herramientas colaborativas e introducen las Arquitecturas de Referencia como planos destilados de mejores prácticas que reducen riesgos de desarrollo, subrayando la necesidad de marcos de este tipo.

b) Compensaciones reactivas vs. tradicionales no definidas: Aunque la literatura ofrece pautas situacionales claras —como recomendar patrones reactivos para escenarios de alta concurrencia y E/S con latencia entre servicios, y patrones tradicionales para cargas lineales o con alta carga de CPU (Kasianchuk, 2024; Mochniej & Badurowicz, 2023)—, se señala la ausencia de un marco de decisión unificado y formalizado que ayude a los arquitectos a evaluar sistemáticamente estas compensaciones (Kumar, 2025).

c**)** Suposiciones iniciales inadecuadas**:** Las pilas tecnológicas conocidas (ej. LAMP) suelen ser insuficientes para requisitos no bloqueantes basados en eventos, lo que lleva a costosas refactorizaciones y deuda técnica (Bass et al., 2012; Dasanayake et al., 2015).

METODOLOGÍA

La investigación siguió un enfoque híbrido inductivo-deductivo en dos fases: una inductiva, que incluyó las subfases de instanciación y validación, y otra deductiva.

Fase inductiva

El estudio partió de observaciones realizadas en un proyecto previo de automatización de flujos de trabajo, donde se constató la capacidad de adaptación del sistema, basado en el principio de que "en lugar de adaptar un flujo de trabajo a un sistema, el sistema se puede personalizar para el flujo de trabajo" (Manko, 2021). Para fundamentar la inspiración en PLC,

ISSN 2594-2298
Prefijo DOI: 10.70417

se estableció una analogía funcional directa entre componentes de hardware y software (véase Tabla 1).

Tabla 1. Analogía funcional PLC-WFMS

PLC (Mohamed et al., 2023)	Equivalente WFMS	Función
Controlador programable y flexible	Sistema configurable y versátil	Adapta contextos industriales y reglas de automatización
CPU	Motor de Flujos de Trabajo (Hollingsworth, 1995)	Interpreta modelos, secuencia actividades, gestiona recursos
Interfaces E/S	Campos interactivos de E/S	Obtienen y transmiten información
Memoria de Programa/Datos	Repositorio de Reglas (Teng et al., 2011)	Almacena reglas y datos operativos
Interfaz de Comunicación	Message Broker / REST API	Integración externa
Interfaz Hombre-Máquina (HMI)	User Interface (UI)	Configuración, notificaciones, control operativo

Se realizó un análisis de los impulsores arquitecturales, donde se definieron:

- Tres escenarios principales: Gestión de tableros de usuario, Automatización del flujo de trabajo, Integración de servicios IoT.
- Atributos de calidad: Desempeño y agilidad de desarrollo.
- Restricción: Implementación con tecnologías JavaScript y/o PHP.

Búsqueda exploratoria de literatura: Se consultaron las bases de datos Scopus, IEEE Xplore, Taylor & Francis y MDPI, complementadas con Google Scholar, para artículos publicados entre 2019 y 2026. Se emplearon combinaciones de palabras clave —como 'workflow automation', 'software adaptability', 'software architecture validation', 'reactive programming', 'real-time technologies' y 'workflow management system architecture'— utilizando operadores booleanos (AND/OR) adaptados a cada motor de búsqueda. Los registros obtenidos (n=111) se gestionaron con Mendeley para la eliminación de duplicados y organización inicial. Posteriormente, se aplicó un cribado por títulos y resúmenes, seleccionando aquellos estudios que abordaban al menos tres de las palabras clave y que incluían validación experimental o estudios de caso, resultando en un conjunto final de 19 artículos para el análisis completo.

ISSN 2594-2298
Prefijo DOI: 10.70417

Conceptualización y diseño: Se sintetizaron los impulsores arquitecturales y la información recabada mediante editor de texto y Mermaid Online (https://www.mermaidonline.live) para diagramas.

Instanciación y validación

Instanciación de la arquitectura: Se utilizó ingeniería de prompts con DeepSeek-V3 (Subahi, 2026) mediante diálogo iterativo para refinar decisiones arquitectónicas (véase Figura 1).



Figura 1. Descripción general del marco PROMPT utilizado

Generación y evaluación de prototipos: Se implementaron prototipos con Visual Studio Code (backend y frontend) y Arduino IDE (firmware para ESP32), utilizando DeepSeek como asistente de programación (véase Figura 2).

Pruebas de comunicación: Se utilizó Postman para pruebas de publicación-suscripción MQTT y Docker para el despliegue del broker Eclipse Mosquitto.

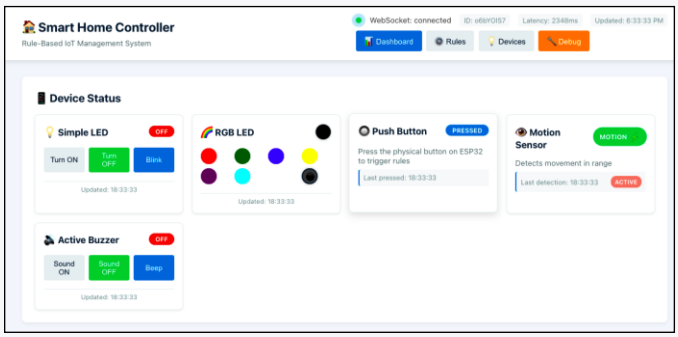


Figura 2. Interfaz web del prototipo de Hogar Inteligente

Validación física: Se empleó un circuito con placa ESP32 configurado en Arduino IDE, incluyendo: LED, sensor PIR, pulsador, LED RGB, zumbador activo, protoboard y jumpers.

Se utilizó un archivo de traza de Chrome DevTools, procesado con DeepSeek-V3 para extraer métricas de rendimiento. Las pruebas se ejecutaron en red local doméstica (localhost:3000), proporcionando un entorno controlado para medir el rendimiento inherente del prototipo.

ISSN 2594-2298
Prefijo DOI: 10.70417

Fase deductiva

Una vez verificada la categoría del sistema (WFMS), los hallazgos se contrastaron con arquitecturas de sistemas WFMS actuales (predominantemente basadas en la nube). Esto permitió:

- Validar las propiedades identificadas inductivamente.
- Conocer otras arquitecturas WFMS para diferentes propósitos.
- Identificar tecnologías WFMS establecidas: Camunda, n8n, AWS Step Functions.

Para la síntesis comparativa entre entornos reactivos (Node.js) y thread-per-request (Apache HTTP Server) en WFMS adaptativos, se utilizó Consensus AI (https://consensus.app) como herramienta de apoyo para extraer hallazgos clave a partir de la consulta: 'Comparison between the implementation of reactive programming in non-blocking environments like Node.js and other technologies like Apache HTTP Server for performance improvement on a Web-Based Adaptive WFMS'. Los resultados fueron contrastados con las fuentes originales para verificar su precisión, y la síntesis final se organizó en torno a los ejes: rendimiento, escalabilidad y complejidad de implementación.

RESULTADOS

La fase inductiva produjo tres resultados principales:

Primero, las observaciones de un proyecto previo de automatización de flujos de trabajo revelaron que el sistema demostró capacidad de adaptación. El mapeo PLC-WFMS reveló equivalencias directas que validaron el modelo PLC como un plan pragmático para un WFMS interoperable.

Segundo, el análisis de impulsores arquitecturales mediante la técnica de caja negra definió tres escenarios funcionales: Gestión de Tableros (monitoreo en tiempo real), Automatización de Flujos de Trabajo (automatización basada en reglas) e Integración de Servicios IoT. Se priorizaron dos atributos de calidad: desempeño (respuesta en tiempo real con baja latencia) y agilidad de desarrollo (iteración rápida). También se estableció una restricción de implementación: las tecnologías deben estar basadas en JavaScript y/o PHP.

Tercero, se identificaron cinco propiedades intrínsecas mediante el proceso inductivo y la revisión de literatura: un motor de flujos de trabajo para gestionar flujos (Hollingsworth, 1995), tecnología en tiempo real para respuesta instantánea a eventos (Jain et al., 2018), mecanismos de publicación-suscripción para distribución de eventos (Negru et al., 2022; Olteanu et al., 2025), programación reactiva como paradigma no bloqueante (Kasianchuk, 2024), y reglas basadas en contexto para evaluación dinámica (Teng et al., 2011).

La subfase de instanciación y validación produjo tres resultados importantes:

Primero, la arquitectura de referencia fue instanciada exitosamente mediante ingeniería de prompts con DeepSeek-V3 (Subahi, 2026) utilizando diálogo iterativo. La arquitectura comprendió (véase Figura 3): una capa de presentación con UI Reactiva y flujos de trabajo basados en reglas (React); una capa de aplicación que incorpora un motor de reglas basado

ISSN 2594-2298
Prefijo DOI: 10.70417

en eventos (patrón Mediador) con API REST sobre Node.js/Express (patrón Enrutador-Acción); una capa de integración que utiliza un puente MQTT (patrón Gateway); y protocolos de comunicación que incluyen HTTP, WebSocket (Socket.IO) y MQTT (Eclipse Mosquitto). La aportación no radica en la creación individual de estos patrones, sino en su organización dentro de una arquitectura de referencia inspirada en PLC para Web-WFMS adaptativos.

Segundo, se implementó un prototipo funcional integrando cinco componentes de hardware con una placa de desarrollo ESP32 (véase Figura 4): LED (GPIO 2), Pulsador (GPIO 4), LED RGB (GPIO 16,17,18), Sensor de Movimiento PIR (GPIO 13) y Zumbador Activo (GPIO 12).

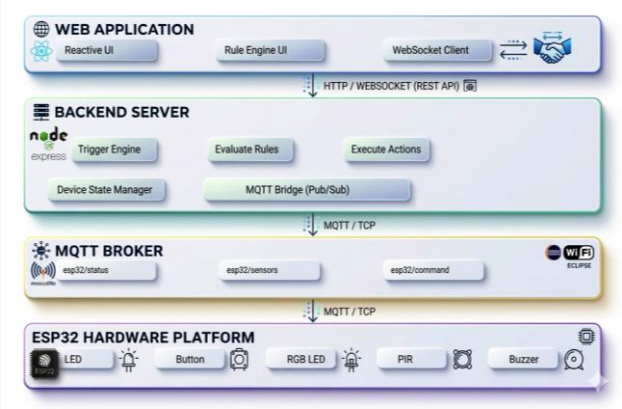


Figura 3. Diagrama de la arquitectura de referencia

Tercero, el prototipo demostró viabilidad técnica mediante la validación de rendimiento a través de un archivo de traza de Chrome DevTools, el cual permitió medir con precisión el comportamiento del sistema. Las métricas obtenidas confirman un rendimiento robusto: la latencia WebSocket presentó un promedio de 1.12 ms (máximo de 56.75 ms), el tiempo de carga mostró una primera pintura con contenido (FCP) de ~360 ms y una primera pintura significativa (FMP) de ~2603 ms, la frecuencia de actualización se mantuvo en ~60 FPS con caídas esporádicas durante interacción intensa, y la ejecución de reglas demostró ser instantánea en su disparo, aunque el procesamiento de acciones puede extenderse entre 100 ms y 500 ms según su complejidad.

La fase deductiva produjo tres resultados relevantes:

Primero, el sistema fue clasificado exitosamente como un Sistema de Gestión de Flujos de Trabajo (WFMS) mediante contrastación con arquitecturas WFMS existentes, predominantemente basadas en la nube. Esta validación confirmó la categoría del sistema y permitió la identificación de patrones arquitectónicos comunes.

Segundo, se llevó a cabo un estudio comparativo de carácter teórico sobre programación reactiva versus subproceso por petición. El análisis confirmó que la programación reactiva ofrece un rendimiento superior en escenarios de alta concurrencia —incluyendo un aumento de 6.5 veces en el volumen de transacciones y tiempos de respuesta un 93 % más rápidos—, pero conlleva una mayor complejidad del código y una curva de aprendizaje más pronunciada (Li & Guo, 2024; Luo et al., 2024; Zbarcela & Tudose, 2024; Zbarcela et al., 2025; Kumar,

ISSN 2594-2298
Prefijo DOI: 10.70417

2025). Si bien resulta ideal para cargas elevadas, las arquitecturas tradicionales de «un hilo por petición» siguen siendo superiores en escenarios de baja carga (Kumar, 2025; Iwanowski & Koziel, 2022).

Tercero, la búsqueda de tecnologías WFMS existentes identificó alternativas maduras para consideración inmediata: n8n, AWS Step Functions y Camunda (véase Tabla 2). Estas tecnologías fueron identificadas como referencias basadas en la literatura, no implementadas en este estudio.

Table 2. Alternativas WFMS consolidadas

Herramienta	Descripción / Enfoque principal	Fuente (con fecha de consulta)
Camunda	Plataforma para orquestar y automatizar procesos de negocio complejos que integran personas, sistemas, agentes de IA y dispositivos, desde el diseño hasta la operación.	Camunda. (s. f.). *Introduction to Camunda 8*. Camunda 8 Docs. Recuperado 18 de mayo de 2026, de https://docs.camunda.io/docs/components/concepts/concepts-overview/
N8n	Herramienta de automatización de flujos de trabajo con licencia *fair-code* que combina capacidades de IA con la automatización de procesos de negocio, permitiendo conectar aplicaciones con APIs.	N8n. (s. f.). *About n8n*. N8n Docs. Recuperado 18 de mayo de 2026, de https://docs.n8n.io/#about-n8n
AWS Step Functions	Servicio de flujo de trabajo visual y sin servidor que ayuda a desarrollar aplicaciones distribuidas, automatizar procesos y orquestar microservicios utilizando los servicios de AWS.	Amazon Web Services. (s. f.). *Orquestación del flujo de trabajo - AWS Step Functions*. Amazon Web Services, Inc. Recuperado 18 de mayo de 2026, de https://aws.amazon.com/es/step-functions/

CONCLUSIONES

La investigación demostró exitosamente que una arquitectura de referencia inspirada en PLC para Web-WFMS adaptativos puede ser diseñada, instanciada mediante ingeniería de prompts y validada a través de un prototipo funcional. El método de ingeniería de prompts demostró ser efectivo para el apoyo a decisiones arquitectónicas, y se proporcionó orientación basada en evidencia para seleccionar entre entornos reactivos y tradicionales. La viabilidad técnica de la solución propuesta fue confirmada mediante un prototipo funcional que logró comunicación en tiempo real con baja latencia.

Se proporcionan nueve recomendaciones para trabajo futuro: (1) ampliar la validación a múltiples dominios, (2) explorar pilas tecnológicas alternativas, (3) realizar pruebas de rendimiento comparativas, (4) evaluar el método de ingeniería de prompts con diferentes

modelos de IA, (5) implementar y evaluar las tecnologías WFMS sugeridas, (6) desarrollar métricas de evaluación formales para arquitecturas instanciadas mediante prompts, (7) realizar estudios longitudinales sobre mantenimiento y escalabilidad, (8) investigar la integración de agentes de IA, y (9) evaluar en diversos entornos de red, incluyendo WAN con latencia variable, múltiples dispositivos simultáneos y condiciones de red adversas..

Se reconocen siete limitaciones: enfoque en un solo dominio, pila tecnológica específica, alcance del prototipo (viabilidad técnica en lugar de producción), especificidad del modelo de IA, caso de validación único, y tecnologías identificadas como referencias en lugar de soluciones implementadas, pruebas realizadas en red local doméstica (LAN).

REFERENCIAS BIBLIOGRÁFICAS

Amazon Web Services. (s. f.). *Orquestación del flujo de trabajo*. Amazon Web Services, Inc. Recuperado 18 de mayo de 2026, de https://aws.amazon.com/es/step-functions/

Bailón-Delgado, D., Mera-Cantos, J., López-Zambrano, J., Pico, J. M., & Meza, J. (2024). API RESTful in workflow engine: Management of administrative processes, applying the workflow reference model. *AIP Conference Proceedings*, *2994*, 030003. https://doi.org/10.1063/5.0193559

Bass, L., Clements, P., & Kazman, R. (2012). *Software Architecture in Practice* (3.ª ed.). Addison-Wesley Professional. https://ptgmedia.pearsoncmg.com/images/9780321815736/samplepages/0321815734.pdf

Camunda. (s. f.). *Introduction to Camunda 8*. Camunda. Recuperado 18 de mayo de 2026, de https://docs.camunda.io/docs/components/concepts/concepts-overview/

Damarowsky, J., Kuehnel, S., Böhmer, M., & Sackmann, S. (2023, 5 noviembre). *A Reference Architecture for a Workflow Management System Front End Designed for Augmented Reality Headsets*. AIS Electronic Library (AISeL). https://aisel.aisnet.org/ecis2023_rp/296

Dasanayake, S., Markkula, J., Aaramaa, S., & Oivo, M. (2015). *Software Architecture Decision-Making Practices and Challenges: An Industrial Case Study* (24.ª ed., pp. 88-97). IEEE. *https://doi.org/10.1109/aswec.2015.20*

Ferencz, K., Domokos, J., & Kovács, L. (2023). Cloud Integration of Industrial IoT Systems. Architecture, Security Aspects and Sample Implementations. *Acta Polytechnica Hungarica*, *21*(4), 7-28. https://doi.org/10.12700/aph.21.4.2024.4.1

Hollingsworth, D. (1995). *Workflow Management Coalition The Workflow Reference Model* (N.º TC00-1003). The Workflow Management Coalition. https://wfmc.org//wp-content/uploads/2022/09/tc003v11.pdf

Iwanowski, S., & Kozieł, G. (2022). Comparative analysis of reactive and imperative approach in Java web application development. *Journal Of Computer Sciences Institute*, *24*, 242-249. https://doi.org/10.35784/jcsi.2999

Jain, R., Sen, A., & Paliwal, K. (2018). *A Comparative Analysis of New Cow Concept of Node.Js With Php* (NCETECE`14 Conference Proceedings, Vol. 2). International journal of engineering research and technology. https://api.semanticscholar.org/CorpusID:174797138

ISSN 2594-2298
Prefijo DOI: 10.70417

Kasianchuk, I. (2024). Orchestration of service-oriented applications with reactive programming techniques. *Technology Audit And Production Reserves*, *4*(2(78)), 24-29. https://doi.org/10.15587/2706-5448.2024.310031

Kumar, K. S. S. (2025). Reactive Programming Paradigms in High-Throughput Distributed Systems. *European Journal Of Computer Science And Information Technology*, *13*(30), 93-103. https://doi.org/10.37745/ejcsit.2013/vol13n3093103

Li, H., & Guo, X. (2024). Performance Analysis of Web Server Side Reactive Programming. *Lecture Notes Of The Institute For Computer Sciences, Social Informatics And Telecommunications Engineering*, 96-108. *https://doi.org/10.1007/978-3-031-53404-1_9*

Luo, J., Zhou, B., Zheng, Y., & Pan, W. (2024). Research on high performance web service construction method based on JavaScript asynchronous programming technique. *Applied Mathematics And Nonlinear Sciences*, *9*(1). https://doi.org/10.2478/amns-2024-2811

Manko, B. A. (2021). The adaptability of Monday.com's app-based software: Discover the company building a flexible business model that adapts to individual company needs. *Journal of Information Technology Teaching Cases*, *12*(2), 156–162. https://doi.org/10.1177/20438869211028855

Mochniej, K., & Badurowicz, M. (2023). Performance comparison of microservices written using reactive and imperative approaches. *Journal Of Computer Sciences Institute*, *28*, 242-247. https://doi.org/10.35784/jcsi.3698

Mohamed, A., Galgal, K., & Mogili, U. (2023). *Programmable Logic Controllers in Flexible Manufacturing System (FMS)*. American Institute of Physics Inc. https://doi.org/10.1063/5.0175867

Murabet, A. E., & Abtoy, A. (2023). Methodologies of the Validation of Software Architectures. *Journal Of Computing Theories And Applications*, *1*(2), 78-85. https://doi.org/10.33633/jcta.v1i2.9332

N8n. (s. f.). *About n8n*. N8n Docs. Recuperado 18 de mayo de 2026, de https://docs.n8n.io/#about-n8n

Negru, C., Musat, G., Colezea, M., Anghel, C., Dumitrascu, A., Pop, F., De Maio, C., & Castiglione, A. (2022). Dependable workflow management system for smart farms. *Connection Science*, *34*(1), 1833-1854. https://doi.org/10.1080/09540091.2022.2083078

Represa, J. G., Larrinaga, F., Varga, P., Ochoa, W., Perez, A., Kozma, D., & Delsing, J. (2023). Investigation of Microservice-Based Workflow Management Solutions for Industrial Automation. *Applied Sciences*, *13*(3), 1835. https://doi.org/10.3390/app13031835

Olteanu, A., Marian, C. G., & Pietraru, R. N. (2025). Smartwatch-Based Monitoring and Alert System for Factory Operators Using Public Cloud Services. *Applied Sciences*, *15*(5), 2806. https://doi.org/10.3390/app15052806

Sahithi, V. V. P. S. (2026). SRWAS: A Web-Based Service Request and Workflow Automation System. *International Journal For Research In Applied Science And Engineering Technology*, *14*(3), 2546-2557. https://doi.org/10.22214/ijraset.2026.78451

Subahi, A. F. (2026). Prompt Engineering for Advancing Software Engineering: Using Generative Language Models in the Development of Domain-Specific Languages. *IEEE Access*, *14*, 22834-22850. https://doi.org/10.1109/access.2026.3663331

Teng, G., Wei, G., Yin, T., & Zhang, H. (2011). *Framework to support application and rule based UI-Control* (Patent N.º US 2011/0078103 A1). Patent Application Publication. https://www.freepatentsonline.com/20110078103.pdf

Vyatkin, V. (2011). IEC 61499 as Enabler of Distributed and Intelligent Automation: State-of-the-Art Review. *IEEE Transactions On Industrial Informatics*, *7*(4), 768-781. https://doi.org/10.1109/tii.2011.2166785

Zbarcea, A., & Tudose, C. (2024). Migrating from Developing Asynchronous Multi-Threading Programs to Reactive Programs in Java. *Applied Sciences*, *14*(24), 12062. https://doi.org/10.3390/app142412062

Zbarcea, A., Tudose, C., & Boicea, A. (2025). Extending the Migration from Asynchronous to Reactive Programming in Java: A Performance Analysis of Caching, CPU-Bound, and Blocking Scenarios. *Applied Sciences*, *16*(1), 90. https://doi.org/10.3390/app16010090