Desarrollo y Evolución de la Arquitectura de Software: Desafíos, Metodologías y Formación Profesional

Development and Evolution of Software Architecture: Challenges, Methodologies and Professional Training.

Mariana Charry Prada, Jesús Ariel González Bonilla

mariana.charry@soy.sena.edu.co, jagonzalezb@sena.edu.co

Tecnólogo en Análisis y Desarrollo de Software, Servicio Nacional de Aprendizaje SENA

Neiva, Colombia

Abstract This article explores various aspects of software development, highlighting the importance of proper planning, tool selection, and architecture to ensure the creation of functional, scalable, and high-quality products. It addresses the impact of design patterns on the efficiency of application development, particularly mobile applications, and emphasizes how their proper implementation can optimize processes and improve the final product. Furthermore, it underscores the relevance of adopting agile methodologies in dynamic environments, prioritizing adaptability and collaboration to adjust to changing requirements and reduce risks.

Regarding software companies, the article highlights the need to remain competitive through transparent practices and robust organizational structures that enable the measurement of goal and deadline compliance. Software architecture is presented as the cornerstone of any project's success, emphasizing its role in defining components and their interrelationships, as well as in implementing models such as the "Waterfall Model" or layered design. Additionally, it delves into the evolution of mobile telecommunications, data protection, and the adaptation to new technologies, such as Service-Oriented Architecture (SOA) and the Internet of Things (IoT).

Finally, the article stresses the importance of training software architects with multidisciplinary skills and the ability to adapt to emerging technologies, which will be crucial for the future of software development.

Keywords Software development, design patterns, agile methodologies, software architecture, software companies, mobile telecommunications.

Resumen Este artículo explora diversos aspectos del desarrollo de software, destacando la importancia de una correcta planificación, elección de herramientas y arquitecturas para garantizar la creación de productos funcionales, escalables y de calidad. Aborda el impacto de los patrones de diseño en la eficiencia del desarrollo de aplicaciones, en particular las móviles, y subraya cómo su correcta implementación puede optimizar procesos y mejorar el producto final. Además, enfatiza la relevancia de adoptar metodologías ágiles en entornos dinámicos, priorizando la adaptabilidad y la colaboración para ajustarse a cambios de requisitos y reducir riesgos.

En cuanto a las empresas de software, el artículo resalta la necesidad de ser competitivas mediante prácticas transparentes y estructuras organizativas robustas que permitan medir el cumplimiento de metas y plazos. La arquitectura de software es presentada como el pilar fundamental para el éxito de cualquier proyecto, destacando su papel en la

definición de componentes y su interrelación, así como en la implementación de modelos como el "Modelo en Cascada" o el diseño en capas. Además, se profundiza en la evolución de la telefonía móvil, la protección de datos y la adaptación a nuevas tecnologías, como la arquitectura orientada a servicios (SOA) y el Internet de las Cosas (IoT).

Finalmente, el artículo resalta la importancia de la formación de arquitectos de software con competencias multidisciplinarias y la capacidad de adaptación a tecnologías emergentes, lo que será crucial para el futuro del desarrollo de software.

Palabras claves Desarrollo de software, patrones de diseño, metodologías ágiles, arquitectura de software, empresas de software, telefonía software.

Introducción

En el inmenso mundo de la programación y desarrollo de software, podemos encontrar aplicaciones móviles, patrones de diseño, una enorme cantidad de lenguajes diferentes, arquitecturas de software, las cuales, cada una tiene su razón de ser, existir e implementar.

Por eso, es importante conocer un poco de cada una, para entender que el software es un arte, una herramienta importante, por lo tanto, es importante tomar un tiempo para cada paso del proceso, la planeación, elección, ejecución y pruebas finales, no realizar las cosas demasiado rápido, porque se pueden presentar especificaciones incompletas, haciendo perder tiempo en la elaboración, sino, al contrario, poder realizar un software bueno y con una funcionalidad correcta sin necesidad de muchos parches, errores o retrasos.

Si se desea formar una empresa, de igual manera es necesario tener ciertos criterios importantes para poder ser una empresa competitiva en todo este campo del desarrollo de software; alguna de las recomendaciones son, buscar que la empresa sea reconocida como una compañía transparentes con los clientes y tener una estructura, la cual debe estar fortalecida con un organismo que permita medir y saber si se están cumpliendo debidamente las metas establecidas en los procesos, productos y proyectos de la compañía, porque es importante, siempre saber y establecer un tiempo para cada elaboración, para poder cumplir con todos los contractos de los clientes.

Los patrones de diseño han mostrado un mayor impacto en el desarrollo de software, ya que ofrecen soluciones comprobadas que facilitan el proceso de creación de aplicaciones. En particular, el desarrollo de aplicaciones móviles puede ser un proceso tedioso debido a que cada aplicación debe pasar por los ciclos de desarrollo para garantizar que cumpla con los atributos de calidad estándar. Una vez que el desarrollador compara los principales patrones de diseño de software, puede implementar la opción más conveniente para el proyecto. De esta manera, se optimiza el proceso y se mejora la calidad del producto final.

Para las empresas dedicadas al desarrollo de software, o aquellas que buscan formarse en este campo, es fundamental contar con ciertos criterios que les permitan ser competitivas, especialmente en países como Colombia, que en los últimos tiempos ha cobrado importancia en el ámbito tecnológico. El artículo destaca diversas recomendaciones, que, al ser comprendidas y aplicadas, permiten a las organizaciones destacar en el mercado. Al respecto, la creación de una arquitectura de software adecuada es esencial, ya que de ella dependen los modelos de desarrollo que guiarán el proceso. Uno de estos modelos es el “Modelo de Cascada”, en el cual la arquitectura del sistema sigue la ingeniería de requerimientos. Este modelo se divide en tres fases: elección del estilo arquitectónico, selección de los patrones de diseño y diseño de los componentes. Estos conceptos son fundamentales para su implementación práctica, y también es importante comprender los estilos arquitectónicos, que son generalizaciones de los patrones de diseño y se consideran una abstracción útil.

Para que un software sea de calidad y funcione correctamente, es crucial evitar acelerar demasiado el proceso, ya que las especificaciones incompletas pueden llevar a re-procesos que consumen tiempo y recursos. En este contexto, las metodologías

ágiles (MA) han transformado la forma en que se desarrollan proyectos de software, enfocándose en la adaptabilidad, la colaboración y el trabajo en equipo. Su énfasis en las primeras etapas del diseño permite ajustar el rumbo del proyecto desde sus inicios, lo cual es esencial en entornos dinámicos donde los requisitos pueden cambiar rápidamente. La flexibilidad de las MA permite a los equipos responder de manera más eficiente a las necesidades del cliente, lo que genera productos finales más alineados con las expectativas del usuario. Este enfoque, que prioriza la interacción constante y la retroalimentación temprana, ha sido clave para reducir riesgos y mejorar la calidad del software.

En este sentido, la arquitectura de software es el pilar sobre el cual se construye el sistema, y su calidad tiene un impacto directo en la capacidad de una organización para alcanzar sus objetivos de negocio. Una buena arquitectura no solo asegura que los sistemas sean funcionales, sino que también facilita la adaptación, el mantenimiento y la escalabilidad a medida que las necesidades evolucionan. Por ello, realizar evaluaciones tempranas y continuas de la arquitectura es crucial. Elegir una arquitectura adecuada desde el principio permite evitar costosos rediseños conforme el sistema crece o se modifica, asegurando la sostenibilidad a largo plazo.

Una de las formas de lograr esta adaptabilidad es a través del diseño en capas, que permite separar claramente las diferentes responsabilidades del sistema, como la capa de acceso a datos, la capa de lógica de negocio y la capa de presentación (interfaz de usuario). Esta separación facilita la modificación y evolución de cada capa de manera independiente, lo que proporciona mayor flexibilidad. Por ejemplo, si se necesita cambiar el tipo de base de datos o actualizar un framework, es posible hacerlo sin afectar la lógica de negocio ni la interfaz de usuario.

La reflexión sobre la gestión de requisitos en las metodologías ágiles frente al enfoque tradicional de la arquitectura de software pone de manifiesto una de las tensiones más fundamentales en el desarrollo de software. Mientras que en un enfoque tradicional o "en cascada" la captura y definición de requisitos es una fase inicial crítica, en las metodologías ágiles, estos requisitos son entendidos como algo más flexible, sujeto a cambios y evolución a lo largo del proyecto. En un enfoque tradicional, una vez que los requisitos son comprendidos y documentados, la arquitectura del software se diseña y ajusta para cumplir con esos requisitos fijos, lo que puede generar limitaciones si surgen cambios a mitad del proceso.

Finalmente, en un entorno cada vez más dominado por la automatización y el uso de la inteligencia artificial, el rol del arquitecto de información se ha intensificado. Este profesional no solo se encarga de estructurar la organización del conocimiento y la información dentro del sistema, sino que también juega un papel crucial para garantizar que el software sea escalable, sostenible y funcional a lo largo del tiempo. En este contexto, los sistemas ciberfísicos (CPS), que integran componentes computacionales y físicos de manera estrecha, requieren sistemas de software robustos que maximicen la eficiencia en el uso de los recursos disponibles. Este desafío se complica cuando se deben considerar factores como la latencia, la confiabilidad, la seguridad y la escalabilidad, aspectos que deben gestionarse con atención para asegurar el éxito del sistema a largo plazo.

La tesis sobre el artículo que aborda el campo de la arquitectura de software, una rama crucial dentro de la Ingeniería de Software, resalta la importancia de la estructura de los sistemas de software complejos. Este campo se dedica a analizar cómo se organizan e interrelacionan los componentes dentro de un sistema para garantizar su efectividad, escalabilidad y flexibilidad. Una de las estrategias clave para lograr estos objetivos es la reutilización de soluciones conocidas, lo que consiste en aplicar patrones, diseños o componentes previamente probados y validados. Esta práctica no solo mejora la eficiencia, sino que también reduce los costos de desarrollo y aumenta la calidad del software. La hipótesis planteada en el artículo, que sostiene que los arquitectos de software reutilizan soluciones conocidas en nuevos diseños, es completamente válida y refleja una realidad observada en la

mayoría de los proyectos de desarrollo de software a gran escala.

La arquitectura de software, en este contexto, se convierte en un componente fundamental dentro del proceso de desarrollo de software, ya que establece las bases para la estructura general del sistema. Su objetivo principal es organizar los diversos elementos que conforman el sistema de manera eficiente, asegurando que interactúen correctamente entre sí y con otros sistemas externos. En este sentido, la arquitectura describe, analiza y formaliza el esquema global del sistema, lo que incluye la distribución de tareas, la asignación de responsabilidades y las relaciones entre los diferentes componentes. Además, la arquitectura de software debe ser cuidadosamente diseñada y documentada. Herramientas como los diagramas de flujo, diagramas de despliegue, casos de uso y diagramas de paquetes son esenciales para representar visualmente y de manera estructurada cómo interactúan los diferentes componentes del sistema, garantizando que todos los elementos del software estén bien definidos y sus interacciones sean claras.

Un aspecto relevante dentro de la arquitectura de software es la programación orientada a objetos (POO), que ha demostrado ser una de las metodologías más poderosas y efectivas para el desarrollo de software, especialmente en sistemas complejos. A través de la revisión de fuentes documentales actualizadas, se ha comprobado que la POO agrupa un conjunto de técnicas y principios que no solo facilitan el desarrollo, sino también el mantenimiento y la evolución de programas con un alto grado de complejidad.

Es importante destacar que el software no se limita simplemente a escribir código. El software es un conjunto integral que incluye programas, procedimientos, estructuras de datos, reglas, documentación y datos asociados. Este enfoque resalta la complejidad que implica crear y mantener sistemas de computación, lo que requiere un enfoque organizado y sistemático para garantizar su funcionalidad, eficiencia y calidad.

La arquitectura de software juega un papel crucial en el éxito de cualquier proyecto de desarrollo de software, y uno de sus beneficios más valiosos es su capacidad para evaluar la viabilidad de una solución desde las primeras fases del proyecto. La arquitectura no debe considerarse solo como un conjunto de estructuras y componentes, sino como un marco estratégico que define cómo se organizará y evolucionará todo el sistema.

En el contexto de las aplicaciones móviles, la evolución de los sistemas operativos móviles hacia soluciones más estables y robustas ha tenido un impacto significativo en la capacidad de los desarrolladores para crear aplicaciones más grandes y complejas. Este avance ha mejorado tanto el rendimiento como la fiabilidad de las aplicaciones, proporcionando una base más sólida sobre la cual construir software innovador y de alta calidad.

Por otro lado, el proceso de descubrimiento de conocimiento en bases de datos ha ganado gran relevancia en las organizaciones actuales, especialmente con el volumen masivo de datos generados y almacenados en sistemas de gestión de bases de datos. Estos grandes volúmenes de datos, provenientes de diversas fuentes y plataformas, contienen patrones, tendencias y relaciones que, cuando son correctamente analizados, pueden aportar un valor significativo para la toma de decisiones, la optimización de procesos y la innovación empresarial.

Finalmente, la creación de un mundo virtual para enseñar Arquitectura de Software a nivel universitario es una iniciativa innovadora que subraya la importancia de integrar nuevas tecnologías en el proceso educativo. Utilizar un entorno virtual para enseñar conceptos complejos como la arquitectura de software no solo hace el aprendizaje más interactivo, sino que también ofrece un espacio dinámico y flexible donde los estudiantes pueden experimentar y aplicar los principios de la arquitectura en un contexto más inmersivo. Esta metodología innovadora refleja cómo la tecnología puede transformar el aprendizaje y facilitar una mejor comprensión de

conceptos clave en el campo del desarrollo de software.

El diseño de software es una de las actividades más críticas en el desarrollo de sistemas, y su importancia no solo radica en la creación de un sistema funcional, sino también en la sostenibilidad y eficiencia a largo plazo del software. Al reflexionar sobre la afirmación de que es más barato diseñar un sistema adecuadamente que simplemente comenzar a programar sin un diseño previo, podemos comprender mejor la justificación económica y estratégica de un buen proceso de diseño. Este enfoque no solo optimiza recursos, sino que también asegura que el sistema sea flexible y fácil de mantener a lo largo del tiempo, evitando costosos retrabajos y asegurando que el software se alinee con las expectativas de los usuarios y las necesidades del negocio.

En este sentido, el artículo sobre arquitectura de software en líneas de producto pone de manifiesto un enfoque muy interesante y estratégico en la ingeniería de software. Al combinar el arte de la definición de arquitecturas con el concepto de líneas de producto, se aborda un desafío importante: cómo diseñar sistemas que sean lo suficientemente flexibles y reutilizables para adaptarse a diferentes contextos y necesidades del mercado, mientras se mantienen eficientes y coherentes a lo largo de su ciclo de vida. Este enfoque, basado en la reutilización y la adaptabilidad, se convierte en una pieza clave para enfrentar los retos del desarrollo de software a gran escala.

A la par de estos avances, la evolución de la telefonía móvil ha transformado profundamente la forma en que interactuamos con el mundo, tanto en términos de comunicación como de acceso a la información. En este proceso, los avances en tecnologías como SOA (Arquitectura Orientada a Servicios) han jugado un papel crucial, al permitir una integración más eficiente y flexible de diversos servicios a través de dispositivos móviles. SOA, como paradigma arquitectónico, se basa en la creación de servicios independientes y reutilizables que pueden ser accedidos por diferentes aplicaciones y sistemas, lo que promueve la interoperabilidad y la escalabilidad.

En el contexto digital y globalizado actual, la protección de datos y el uso de los datos personales son elementos esenciales para garantizar la privacidad y la seguridad de los individuos. En países como Ecuador y Colombia, donde las normativas sobre protección de datos personales están claramente definidas, el marco legal se orienta a preservar los derechos fundamentales de los ciudadanos frente al uso de su información personal. La reflexión que surge en torno a este tema involucra tanto los principios legales y éticos como los desafíos que presentan los avances tecnológicos y la globalización, haciendo imperativa la creación de arquitecturas de software que respeten la privacidad y la seguridad.

Por otro lado, una de las principales ventajas de las aplicaciones web es su capacidad de operar de manera independiente del sistema operativo del usuario. A diferencia de las aplicaciones tradicionales, que deben ser adaptadas y distribuidas para cada plataforma (Windows, macOS, Linux, etc.), las aplicaciones web se ejecutan en navegadores. Esto les permite ser accesibles desde cualquier dispositivo con acceso a Internet, lo que aumenta enormemente su alcance y flexibilidad. La independencia del sistema operativo también facilita la adopción de nuevas tecnologías, ya que los usuarios no se ven limitados por las especificaciones o compatibilidades de sus dispositivos.

En este contexto, los patrones de diseño se convierten en herramientas fundamentales para mantener la calidad y modularidad del código. Aunque no existe un patrón "superior" universalmente, el verdadero desafío radica en identificar cuál se adapta mejor a las necesidades del proyecto en particular. Este enfoque flexible y adaptativo permite a los desarrolladores crear soluciones más eficientes, manteniendo la integridad y la claridad del código a largo plazo. Al final, la habilidad de seleccionar el patrón adecuado se convierte en un arte que mejora tanto

la mantenibilidad como la escalabilidad del software.

Sin embargo, en un mundo donde los tiempos de desarrollo son cada vez más cortos y las expectativas sobre las funcionalidades de los sistemas son cada vez más altas, los métodos tradicionales de desarrollo de software tienden a volverse insuficientes. Los equipos de desarrollo se enfrentan al reto de crear soluciones más rápidas sin comprometer la calidad o la capacidad de adaptación. Aquí, la reutilización de componentes y el bajo acoplamiento entre los diferentes módulos o servicios permiten crear arquitecturas de software más rápidas y eficientes.

La evolución de la arquitectura de software y su representación en los últimos años apunta a una necesidad cada vez más clara: encontrar un equilibrio entre la formalidad y la accesibilidad. En el pasado, los diagramas de "cajas y líneas" eran, y en muchos casos siguen siendo, una herramienta visual útil para representar componentes y relaciones dentro de un sistema. Sin embargo, su simplicidad también ha sido uno de sus límites con el crecimiento de los sistemas de software y su integración en entornos más complejos y cambiantes. Esta transición hacia una representación más dinámica y detallada es fundamental para manejar sistemas más sofisticados.

Finalmente, la evolución hacia el Internet de las Cosas (IoT) representa una de las transformaciones tecnológicas más fascinantes y disruptivas de la era moderna. Al conectar objetos cotidianos a la red, estamos redefiniendo no solo la interacción con el entorno, sino también el concepto mismo de "inteligencia". Los dispositivos ya no son simples herramientas, sino que se convierten en elementos autónomos capaces de percibir, procesar y reaccionar a eventos del mundo real, todo ello a través de la conectividad y la interacción. Este cambio está impulsando nuevas demandas y desafíos para la arquitectura de software, que debe ser capaz de gestionar y analizar la enorme cantidad de datos generados por estos dispositivos.

En conclusión, la evolución de la arquitectura del software refleja el creciente reconocimiento de su papel fundamental en el éxito de los sistemas software. Tradicionalmente, la arquitectura se entendía como una fase del diseño dentro del proceso global de desarrollo, con énfasis en la creación de un conjunto de componentes y la definición de sus relaciones. Sin embargo, al aumentar la complejidad de los sistemas, el enfoque ha pasado a ser mucho más holístico, reconociendo que la arquitectura no solo debe ser una fase técnica, sino también un campo de estudio y práctica en sí mismo, esencial para la creación de sistemas eficientes, sostenibles y adaptables.

La formación de un arquitecto de software es, efectivamente, un desafío complejo debido a la naturaleza multidisciplinaria de esta profesión y la rápida evolución de la tecnología. Como señalas, las universidades y las instituciones de educación superior habituales no pueden ofrecer la experiencia práctica completa que los arquitectos de software necesitan para enfrentar los desafíos reales de la industria. En este sentido, es crucial identificar y definir las competencias mínimas que debe tener un arquitecto de software, ya que estas habilidades no solo deben estar alineadas con las exigencias del mercado laboral, sino también con la capacidad de adaptación constante a nuevas tecnologías y metodologías. Además, la formación de un arquitecto de software debe incluir la capacidad de trabajar en equipo, la gestión de proyectos, la toma de decisiones estratégicas y el dominio de diversas herramientas y marcos arquitectónicos, lo que requiere una combinación de conocimientos técnicos y habilidades interpersonales.

El desarrollo de software es una disciplina compleja y multifacética que involucra una gran cantidad de factores técnicos, humanos y organizacionales. Como bien menciona, las metodologías tradicionales han sido fundamentales en la estructuración del proceso de desarrollo, proporcionando un marco claro que ayuda a gestionar las actividades, los artefactos y las herramientas. Este enfoque ha demostrado ser eficaz en muchos contextos, especialmente cuando se trata de proyectos grandes y bien definidos,

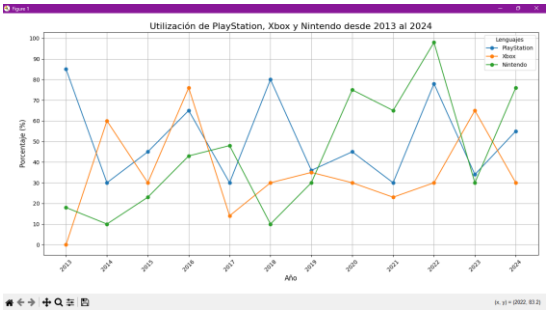
donde el control y la predictibilidad son esenciales. Sin embargo, a medida que los proyectos se vuelven más dinámicos y los requisitos cambian constantemente, la rigidez de estos enfoques se ve desafiada, abriendo espacio para nuevas metodologías que permiten mayor flexibilidad y adaptación.

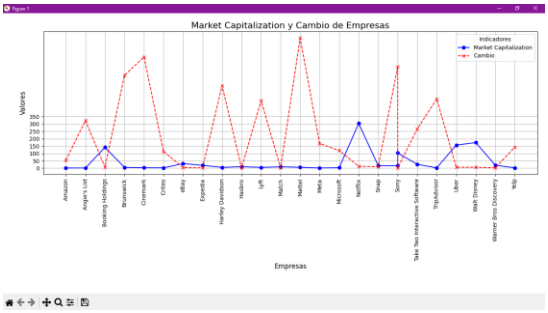
La reflexión sobre el diseño de arquitecturas de software como un proceso creativo, y la falta de herramientas específicas para la reutilización, revela varias cuestiones importantes. En primer lugar, el hecho de que la creación de arquitecturas de software no esté estandarizada resalta la naturaleza profundamente personalizada y subjetiva de este proceso. Cada arquitecto, dependiendo de su experiencia y enfoque, puede abordar un problema de una manera única. Esto es un reflejo de la complejidad inherente a la arquitectura de software, donde los desafíos son tan diversos y cambiantes que un enfoque rígido o universal no suele ser efectivo. Esta flexibilidad es vital para poder adaptar las soluciones a las necesidades particulares de cada proyecto, pero también plantea desafíos para la educación y la estandarización de prácticas.

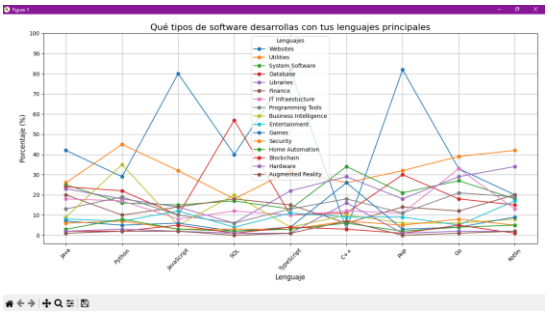
La evolución de las metodologías de Desarrollo de Software (DS) refleja un aprendizaje colectivo que nace de los fracasos y dificultades en los primeros días de la informática. En sus inicios, el desarrollo de software carecía de las estructuras y prácticas que hoy consideramos esenciales. Esto daba lugar a un proceso artesanal, donde la creatividad y las habilidades individuales eran claves, pero también a una gran incertidumbre en cuanto a la calidad, el tiempo y el presupuesto. Los proyectos eran propensos al fracaso porque no se contaban con los marcos necesarios para garantizar que los objetivos del cliente se cumplieran. A medida que la industria avanzó, se fue reconociendo la necesidad de sistemas más estructurados, lo que llevó al desarrollo de metodologías más formales como el modelo en cascada, las metodologías ágiles, y más recientemente, la integración de enfoques híbridos que buscan combinar lo mejor de ambos mundos: control y flexibilidad.

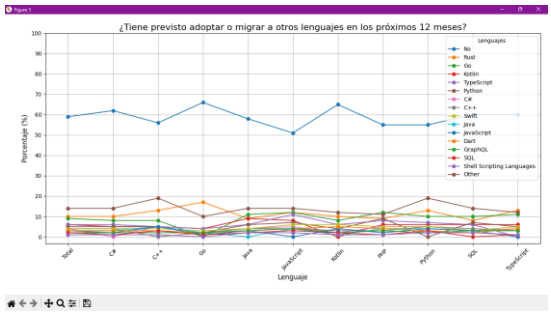
Este aprendizaje continuo ha sido fundamental para mejorar la eficacia del desarrollo de software, y se refleja en la evolución de las herramientas, los procesos y las competencias necesarias para enfrentar los desafíos actuales. Las metodologías ágiles, por ejemplo, surgieron precisamente para abordar las limitaciones de los enfoques tradicionales en entornos dinámicos y con cambios frecuentes, permitiendo a los equipos adaptarse de manera más eficiente a los requisitos cambiantes del cliente.

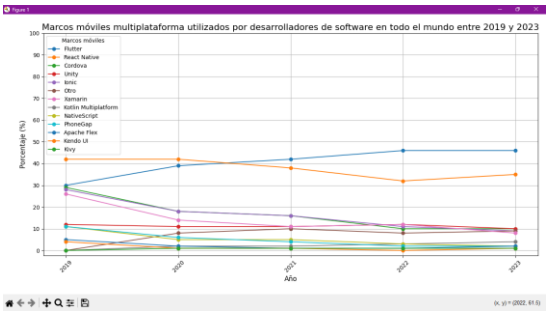
Resultados

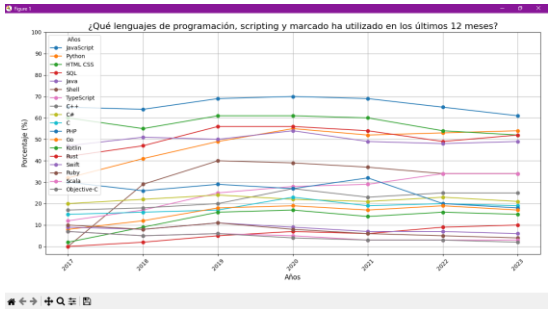


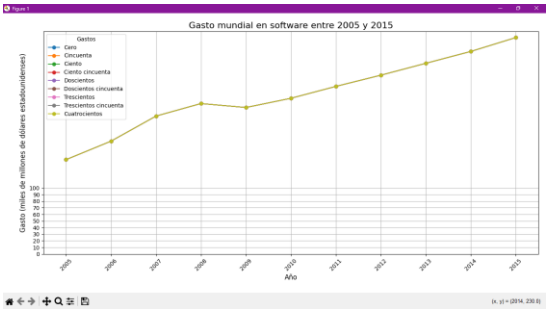


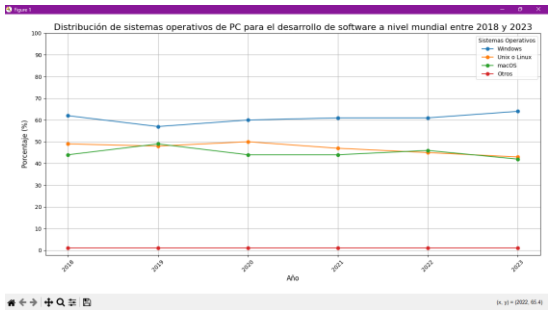


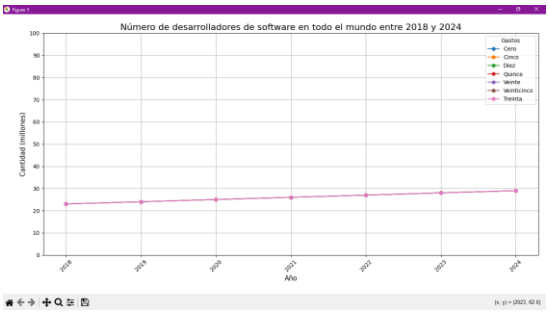












REFERENCIAS

[1] Abanto Cruz, J. A., & Gonzales Ramírez, O. F. (2019). Análisis comparativo de patrones de diseño de software para el desarrollo de aplicaciones móviles de calidad: Una revisión sistemática de la literatura.

[2] Londoño, L. F. L. (2005). Recomendaciones para la Formación de una Empresa de Desarrollo de Software Competitiva en un País como Colombia. Avances en Sistemas e Informática, 2(1), 41-52.

[3] Cristiá, M. (2008). Introducción a la Arquitectura de Software. Research-Gate.[Online]. Recuperado de: https://www.

researchgate. net/publication/251932352 Introduccion a la Arquitectura de Software.

[4] Jimenez-Torres, V. H., Tello-Borja, W., & Rios-Patiño, J. I. (2014). Lenguajes de patrones de arquitectura de software: una aproximación al estado del arte. Scientia et technica, 19(4), 371-376.

[5] Navarro, M.E., Moreno, M.P., Aranda, J., Parra, L., Rueda, J.R., & Pantano, J.C. (2017, September). Integración de arquitectura de software en el ciclo de vida de las metodologías ágiles. In XIX Workshop de Investigadores en Ciencias de la Computación (WICC 2017, ITBA, Buenos Aires).

[6] Sanabria, E.R., & Rodríguez, S.V. (2021). Evaluación de una arquitectura de software. Prospectiva, 19(2).

[7] Cardacci, D. G. (2015). Arquitectura de software académica para la comprensión del desarrollo de software en capas. (No. 574). Serie Documentos de trabajo.

[8] Navarro, M.E., Moreno, M.P., Aranda, J., Parra, L., Rueda, J.R., & Rueda, J.R. (2018). Arquitectura de software en el proceso de desarrollo ágil: una perspectiva basada en requisitos significantes para la arquitectura. In XX Workshop de Investigadores en Ciencias de la Computación (WICC 2018, Universidad Nacional del Nordeste).

[9] Moyares, Y.; & Lorenzo, D.B.; (2021). La Arquitectura de información (AI) en el proceso de desarrollo de software. Bibliotecas Anuales de investigación, 6, 97- 102.

[10] Ting, J.S.(2011). Arquitectura de software par a los actuales sistemas ciberfísicos. Revista Ingenierías USBmed, 2(1), 29.

[11] CUESTA QUINTERO, C.E.(2002). Arquitectura de software dinámica basada en reflexión (Doctoral dissertation, Universidad de Valladolid).

[12] Carignano, M.C.(2016, June). Representación y razonamiento sobre las decisiones de diseño de arquitectura de software. In XVIII Workshop de Investigadores en Ciencias de la computación.) (WICC 2016, entre Ríos, Argentina).

[13] Martín, Y.E(2012). Arquitectura de Software. Arquitectura orientada a servicios. Serie científica de la Universidad de las Ciencias informáticas, 5(1), 1-10.

[14] Cárdenas- Gutiérrez, J.A., Barrientos-Monsalves, E, S, & Molina- Salazar L.(2022). Arquitectura de software para el desarrollo de herramientas tecnológicas de costos, presupuestos y programación de obra. I+D Revista de investigaciones, 17(1), 89-100.

[15] Vera, J.B.V.(2023). Arquitectura de software con programación orientada a objetos. Polo del conocimiento: Revista científico-profesional, 8(12), 1497-1508.

[16] Mamani, S.M.(2023). La importancia de la Arquitectura de software. In Memorias del Congreso Nacional de Informática (No.1, pp.41-50).

[17] Nuñez, G. & Camacho, E. (2004). Arquitectura de software. Guía de estudio.

[18] Granada, E.Z., Rodríguez, L.E.S., Montoya, C.E.G., & Uribe, C.A.C. (2014). Arquitectura de software para entornos móviles. Revista de Investigaciones Universidad del Quindío, 25 (1), 20-27.

[19] De Abadía, M.E.V., Cuevas, C.M.G., González, M.E.M., & Bueno, J.A.A., (2001). Arquitectura de software para descubrimientos de conocimiento. Energía y Computación, 10 (1), 6-13.

[20] Casado Manzanero, V. (2009). Diseño de un mundo virtual para la enseñanza de arquitectura de software.

[21] Cristián, M. (2021). Una teoría para el diseño de software.

[22] Romo Moreno, A. (2009). Definición de Arquitectura para una línea de productos de software.

[23] Santos, L. M., Rico, D. W., & Rincón, A. A. (2009). Servicios web en telefonía celular. Scientia et technica, 15(42), 363-368.

[24] Quishpe, M. V., Moreano, J. C., Guanoluisa, A. G., & Atavallo, C. C. (2023). Protección de Datos Personales en Ecuador y Colombia: Principios, Ética y Desafíos Actuales. Revista

Científica de Informática ENCRIPTAR-ISSN: 2737-6389., 6(11), 19-34.

[25] Trejos Arroyave, M. H., & Zamora Cardona, D. F. (2012). Criterios de evaluación de plataformas de desarrollo de aplicaciones empresariales para ambientes web.

[26] Alvarez, O. D. G., Larrea, N. P. L., & Valencia, M. V. R. (2022). Análisis comparativo de Patrones de Diseño de Software. Polo del Conocimiento: Revista científico-profesional, 7(7), 2146-2165.

[27] Romero, P. Á. (2006). Arquitectura de software, esquemas y servicios. Ingeniería Industrial, 27(1), 1.

[28] Gil, S. V. H. (2003). Representación de la arquitectura de software usando UML. Sistemas y Telemática, 1(1), 63-75.

[29] Segura, A. A. (2016). Arquitectura de software de referencia para objetos inteligentes en internet de las cosas. Archivo de la Revista Latinoamericana de Ingeniería de Software, 4(2), 73-110.

[30] Linero, J. M. T. (1996). Arquitectura software de sistemas abiertos. In II Jornadas de informática. Actas: Almuñécar (Granada), 15 al 19 de julio 1996 (p. 3).

[31] Yépez, W. L. P., Alegría, A. F. S., Bandi, A., & Alegría, J. A. H. (2024). Visión de las competencias en arquitectura de software integrando las perspectivas de la industria y la academia. REVISTA COLOMBIANA DE TECNOLOGIAS DE AVANZADA (RCTA), 1(43), 9-23.

[32] Canós, J. H., Letelier, P., & Penadés, M. C. (2003). Metodologías ágiles en el desarrollo de software. Universidad Politécnica de Valencia, Valencia, 1-8.

[33] Carignano, M. C., Gonnet, S. M., & Leone, H. P. (2016, November). RADS: una herramienta para reutilizar estrategias en diseños de arquitecturas de software. In Simposio Argentino de Ingeniería de Software (ASSE 2016)-JAIIO 45 (Tres de Febrero, 2016).

[34] Gamboa, J. Z. (2018). Evolución de las Metodologías y Modelos utilizados en el Desarrollo de Software. INNOVA Research Journal, 3(10), 20-33.