Resumen y reflexión de artículos relacionados con “patrones de diseño” y “arquitectura de software”

Aura María Fierro Fierro

Instructor:

Jesús Ariel González Bonilla

Centro de la industria, la empresa y los servicios

Análisis y desarrollo de software

Neiva-Huila

15 de noviembre del 2024

Soy Aura María Fierro, estudiante del SENA en la tecnología de Análisis y Desarrollo de Software. Apasionada por el mundo del desarrollo y el arte, he encontrado en la programación una herramienta para crear soluciones innovadoras. Además, cuento con un título de Técnico en Multimedia, lo que me ha permitido combinar mi creatividad con habilidades técnicas. Mi objetivo es seguir creciendo en este campo, aprendiendo nuevas tecnologías y mejorando mis capacidades para contribuir a la transformación digital. Siempre busco aprender y aplicar mis conocimientos en proyectos que generen un impacto positivo.

ARTICULO 1

Arquitectura de software para el desarrollo de herramienta Tecnológica de Costos, Presupuestos y Programación de obra.

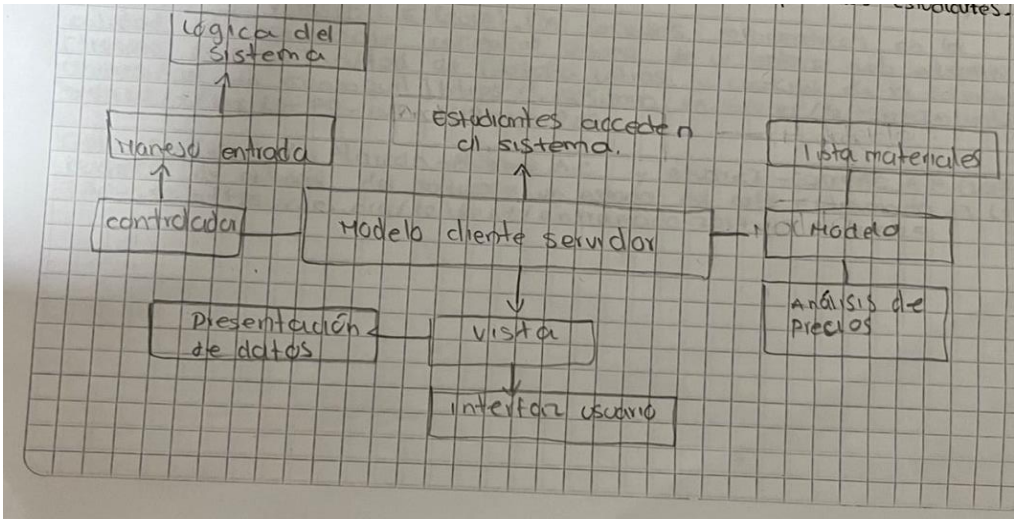
Ese artículo presenta el desarrollo de un software educativo diseñado específicamente para la gestión de costos, presupuestos y la programación de obras en el programa de Ingeniería Civil de la Universidad Francisco de Paula Santander. Este proyecto tiene como objetivo actualizar y modernizar los métodos tradicionales de enseñanza al incorporar una herramienta tecnológica accesible, interactiva y práctica que facilite el aprendizaje y la comprensión de conceptos complejos por parte de los estudiantes. Basado en el modelo Vista-Controlador, el software incluye funcionalidades como la creación y gestión de presupuestos, el cálculo detallado de costos, la planificación de recursos materiales y humanos, y la optimización de la administración de los mismos.

Reflexión:

La reflexión que deja este artículo es que proporcionar herramientas tecnológicas en la educación puede transformar significativamente la formación de futuros profesionales. Los estudiantes no solo adquieren conocimientos teóricos, sino que también desarrollan habilidades prácticas esenciales para enfrentarse a los retos del mundo laboral. Esta iniciativa demuestra cómo la tecnología puede superar barreras económicas, hacer más eficiente el aprendizaje y garantizar una formación más accesible y equitativa.

Además, la implementación del patrón MVC en el desarrollo de software resalta cómo las buenas prácticas de programación no solo mejoran la estructura técnica, sino que también facilitan la comprensión y el manejo del software por parte de los estudiantes. Este enfoque evidencia la importancia de integrar metodologías ágiles y organizadas, fomentando tanto la innovación en la enseñanza como la preparación de los futuros ingenieros para afrontar los desafíos tecnológicos y profesionales de su campo.

Diagrama:



Bibliografía:

ARTICULO 2

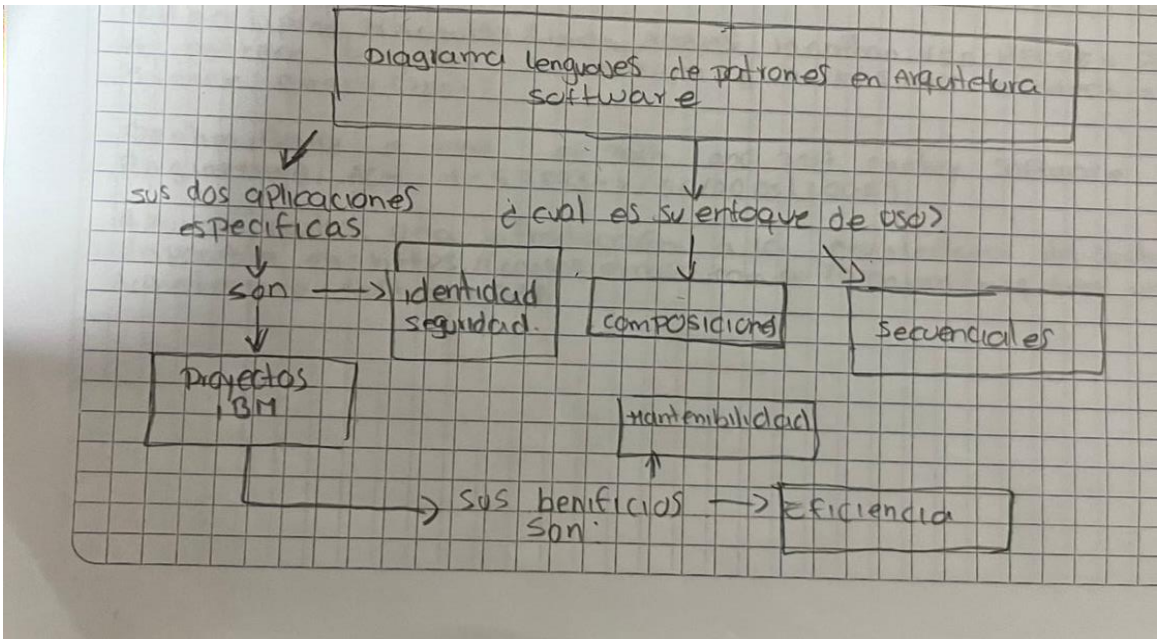
Lenguajes de Patrones de Arquitectura de Software: Una Aproximación Al Estado del Arte.

El artículo examina la evolución de los lenguajes de patrones dentro de la arquitectura de software, partiendo de los conceptos de Christopher Alexander en arquitectura física y adaptados luego al diseño de sistemas informáticos. Se destaca cómo estos patrones han facilitado el diseño modular y reutilizable, mejorando la adaptabilidad y calidad de los sistemas. Los lenguajes de patrones permiten resolver problemas comunes de diseño y son utilizados ampliamente en sectores industriales, como en IBM para arquitecturas e-business, y en áreas específicas como la administración de identidades y seguridad de la información. Los autores revisan múltiples enfoques, como la composición de patrones, historias, secuencias y colecciones, mostrando cómo estos métodos ayudan a estructurar y documentar soluciones eficientes. También se mencionan aplicaciones exitosas en la industria y el impacto de estos lenguajes en la ingeniería de software actual, permitiendo resolver problemas arquitectónicos complejos de manera ágil.

Reflexión:

La reflexión derivada del artículo "Lenguajes de Patrones de Arquitectura de Software: Una Aproximación al Estado del Arte" destaca la importancia de los lenguajes de patrones en la ingeniería de software moderna. Estos patrones no solo facilitan la resolución de problemas recurrentes en el diseño arquitectónico, sino que también promueven un enfoque estructurado y compartido en el equipo de desarrollo, generando una base común de comprensión. Al estandarizar buenas prácticas y soluciones, los lenguajes de patrones contribuyen a construir sistemas más robustos y adaptables, permitiendo que los arquitectos de software aborden problemas complejos con herramientas comprobadas. Además, la versatilidad de estos patrones en diversos contextos y dominios resalta su valor en la creación de soluciones escalables y sostenibles. En un ámbito de constante cambio tecnológico, los lenguajes de patrones se presentan como un recurso indispensable para mantener la cohesión, eficiencia y calidad en el desarrollo de software.

Diagrama:



Bibliografía:

Jimenez-Torres, Victor Hugo; Tello-Borja, Wilman; Rios-Patiño, Jorge Iván Lenguajes de Patrones de Arquitectura de Software: Una Aproximación Al Estado del Arte Scientia Et Technica, vol. 19, núm. 4, diciembre, 2014, pp. 371-376 Universidad Tecnológica de Pereira Pereira, Colombia

ARTICULO 3

Implementación de una Arquitectura de Software

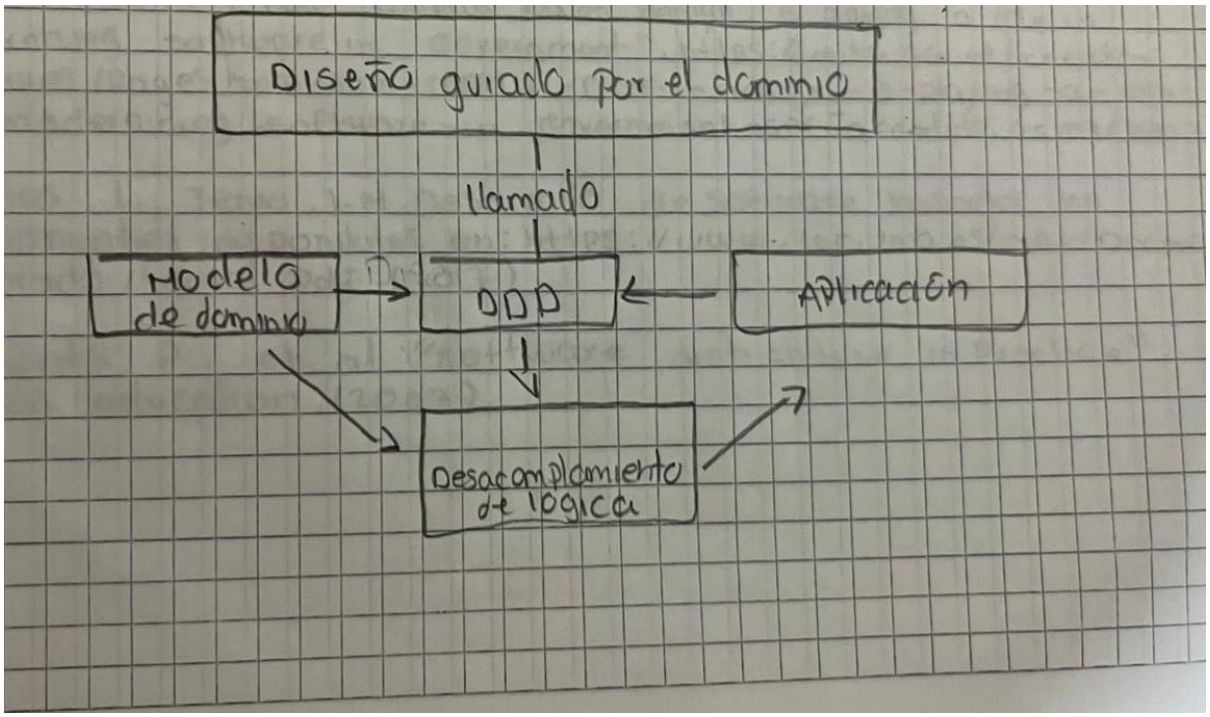
guiada por el Dominio.

Este artículo explora la implementación de una arquitectura de software guiada por el dominio (Domain-Driven Design o DDD) en la creación de sistemas más eficientes y sostenibles. El enfoque de diseño dirigido por el dominio (DDD) permite crear modelos que se centran en los aspectos esenciales del negocio, manteniendo la complejidad técnica en capas independientes. Además, se plantea la integración de arquitecturas limpias y hexagonales para desacoplar las capas del sistema, facilitando la evolución y el mantenimiento del código. En el caso de estudio, se aplica DDD para desarrollar una plataforma de empleo que conecta a usuarios demandantes y oferentes de servicios, utilizando un modelo que mejora la comunicación entre los diferentes actores del sistema. El modelo de dominio garantiza que los componentes del sistema reflejen el lenguaje y reglas de negocio, proporcionando una base sólida y coherente para el desarrollo. La arquitectura hexagonal facilita la prueba y modificación de componentes individuales, adaptándose a las necesidades del negocio sin afectar el núcleo del sistema. Este trabajo demuestra la viabilidad del DDD y la arquitectura hexagonal para enfrentar la complejidad en el diseño de sistemas empresariales.

Reflexión:

La aplicación del enfoque DDD en la ingeniería de software no solo permite crear sistemas centrados en el dominio del negocio, sino que también establece un cambio de paradigma en cómo se percibe y desarrolla el software. Más allá de la técnica, el DDD invita a construir una comprensión común entre desarrolladores y expertos en el negocio, lo cual enriquece la comunicación y alinea las expectativas con los resultados. La arquitectura hexagonal, al permitir un desacoplamiento entre la lógica del negocio y la infraestructura, garantiza que el sistema sea más adaptable y menos dependiente de las tecnologías específicas. Esto refleja una evolución hacia sistemas más sostenibles y robustos, donde el foco se sitúa en los aspectos estratégicos del negocio, manteniendo la tecnología como un soporte adaptable.

Diagrama:



Bibliografía:

Cambarieri M, Difabio F, García Martínez N. ***Implementación de Una Arquitectura de Software Guiada Por El Dominio*****. Accessed November 19, 2024.**

https://n9.cl/dch02

ARTICULO 4

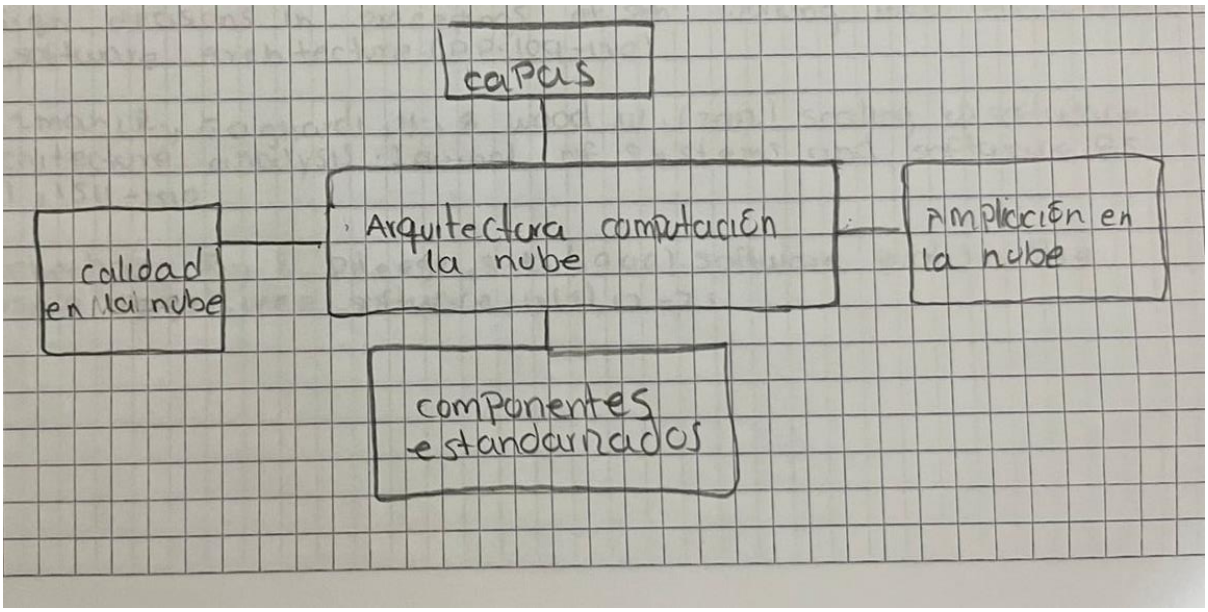
Modelado y Verificación de Patrones de Diseño de Arquitectura de Software para Entornos de Computación en la Nube.

Este artículo describe un enfoque para diseñar y verificar patrones de arquitectura de software orientados a entornos de computación en la nube. La propuesta se basa en un metamodelo de componentes arquitectónicos que facilita la creación de aplicaciones web robustas, organizando los elementos esenciales de diseño. La estructura incluye una herramienta de modelado gráfico que permite a los arquitectos aplicar patrones de diseño y evaluar su aplicación mediante mecanismos de verificación. Esta metodología se destaca por su capacidad de crear arquitecturas de software eficientes y adaptables, organizadas en capas que separan funciones de aplicación e infraestructura. Este enfoque busca optimizar la productividad en el desarrollo y reducir la complejidad del diseño en entornos de nube, beneficiando la calidad y consistencia de las soluciones de software mediante una mayor estandarización y control de las decisiones arquitectónicas.

Reflexión:

Este artículo deja como reflexión la importancia de una metodología estructurada para el diseño de arquitecturas en la nube, donde la separación de funciones y la estandarización de componentes facilitan la adaptación y escalabilidad de los sistemas. El uso de herramientas de modelado y verificación permite un control más riguroso sobre los patrones aplicados, lo cual no solo mejora la calidad del software, sino que también asegura que las decisiones arquitectónicas estén alineadas con los objetivos de negocio y funcionalidad. Esta propuesta sugiere que los arquitectos de software en la nube deben manejar tanto conocimientos técnicos como habilidades de diseño estratégico para enfrentar los desafíos únicos de este entorno, adaptándose rápidamente a las demandas del mercado y de los usuarios.

Diagrama:



Bibliografía:

Blas MJ, Leone H, Gonnet S. Modelado y Verificación de Patrones de Diseño de Arquitectura de Software para Entornos de Computación en la Nube. ***RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação*****. 2019;(35):1-17. doi:https://doi.org/10.17013/risti.35.1-17**

ARTICULO 5

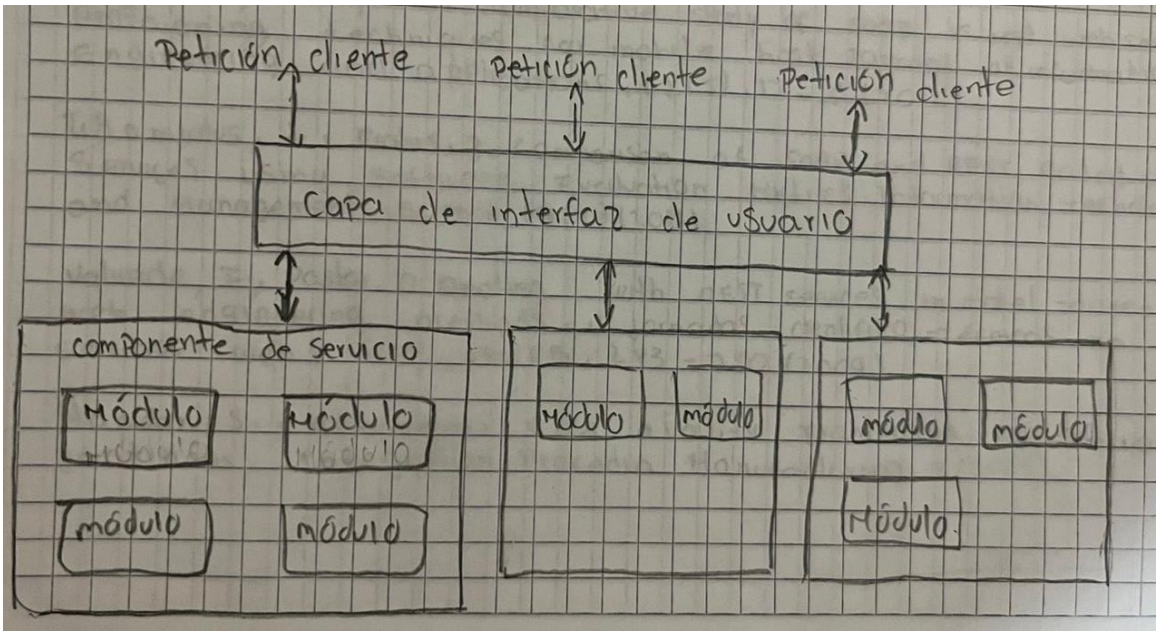
Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web.

Este artículo aborda el uso de una arquitectura de software basada en microservicios como alternativa al modelo monolítico tradicional, especialmente en el contexto de desarrollo de aplicaciones web en la Asamblea Nacional del Ecuador. La arquitectura de microservicios permite separar las funciones en servicios independientes, mejorando la escalabilidad, mantenibilidad y el tiempo de despliegue de nuevas funcionalidades. En lugar de empaquetar toda la funcionalidad en una única unidad, los microservicios funcionan de forma autónoma, cada uno con su lógica de negocio y su base de datos propia, facilitando la administración descentralizada y la tolerancia a fallos. Se mencionan topologías comunes, como la basada en API REST, la centralizada de mensajería y la estructura en capas, cada una adaptada a diferentes necesidades. Este modelo facilita actualizaciones y minimiza el impacto en los usuarios al desplegar solo los servicios específicos requeridos. La investigación concluye que esta arquitectura permite un desarrollo más ágil y adecuado a las necesidades empresariales actuales.

Reflexión:

Este artículo deja como reflexión la importancia de adoptar enfoques modernos y flexibles como la arquitectura de microservicios en el desarrollo de software. A medida que las organizaciones crecen y se enfrentan a demandas cada vez más complejas, una estructura modular y desacoplada permite responder de manera más eficiente a los cambios y reducir el impacto de fallos en el sistema. La arquitectura de microservicios no solo mejora la escalabilidad, sino que también aporta mayor resiliencia al permitir que los componentes del sistema se desplieguen y administren de forma independiente. Esta flexibilidad resulta esencial en un contexto donde la adaptación rápida y la continuidad de las operaciones son claves para satisfacer las necesidades del negocio y del usuario final.

Diagrama:



Bibliografía:

a Asamblea Nacional del Ecuador, Coordinación General de Tecnologías de la Información y Comunicación, Piedrahita 5-21 y Av. Gran Colombia, Pichincha, Ecuador daniel.lopez@asambleanacional.gob.ec b Universidad Técnica del Norte, Instituto de Posgrados, Av. 17 de Julio 5-21 Gral. José María Cordova, Imbabura, Ecuador eamaya@utn.edu.ec

ARTICULO 6

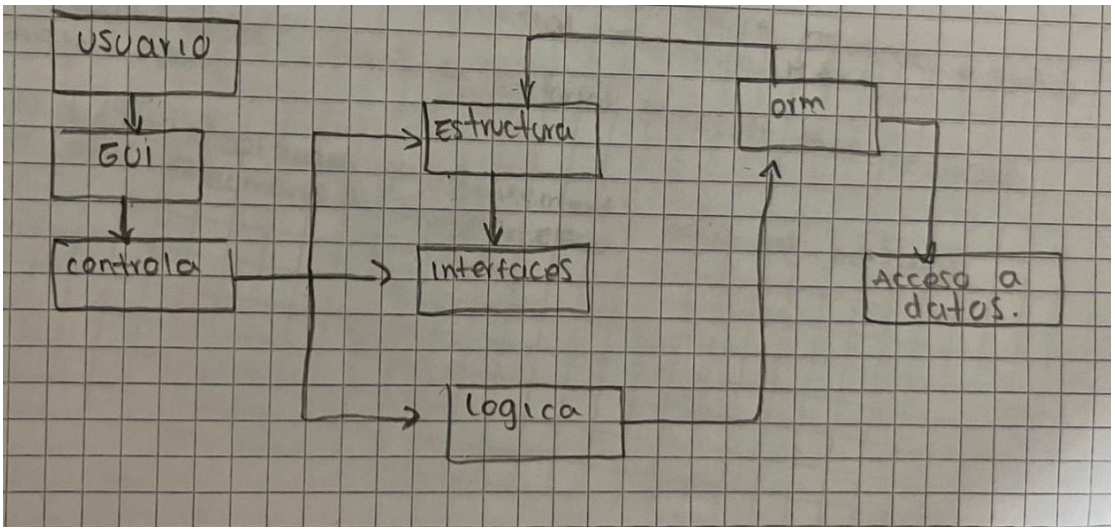
Arquitectura de software académica para la comprensión del desarrollo de software en capas.

El artículo presenta una arquitectura de software en capas diseñada específicamente para el ámbito educativo, cuyo objetivo es ayudar a los estudiantes a comprender los fundamentos de un desarrollo de software estructurado y organizado. Esta arquitectura se basa en una división clara en capas: la capa de interfaz gráfica (GUI), que maneja la interacción visual con el usuario; la capa de control, que se encarga de gestionar la comunicación entre la interfaz y la lógica; la capa de lógica de negocio, donde reside la funcionalidad central del sistema; y la capa de acceso a datos, que gestiona la conexión con las bases de datos y el almacenamiento de información. La organización en capas facilita la modularidad del sistema, lo cual permite realizar cambios en una capa sin afectar a las demás, lo que es ideal para el mantenimiento y la escalabilidad a largo plazo.

Reflexión:

Este artículo deja como reflexión la importancia de enseñar el desarrollo de software a través de una arquitectura en capas, un enfoque que facilita el aprendizaje de conceptos fundamentales de organización y modularidad. La arquitectura en capas permite a los estudiantes comprender cómo dividir un sistema en componentes independientes, lo cual es esencial para el mantenimiento y la escalabilidad de software en el mundo real. Además, les proporciona una base sólida para manejar la complejidad de sistemas grandes, donde cada capa cumple una función específica y reduce la dependencia entre componentes. Esta metodología de enseñanza sugiere que los futuros profesionales estarán mejor preparados para adaptarse a cambios y optimizar el rendimiento de sistemas en entornos diversos. En un campo donde la tecnología y las demandas cambian rápidamente, una estructura clara y modular como esta asegura que los estudiantes desarrollen habilidades técnicas y estratégicas valiosas, necesarias para construir y mantener sistemas de alta calidad.

Diagrama:



Bibliografía:

Suggested Citation: Cardacci, Darío G. (2015) : Arquitectura de software académica para la comprensión del desarrollo de software en capas, Serie Documentos de Trabajo, No. 574, Universidad del Centro de Estudios Macroeconómicos de Argentina (UCEMA), Buenos Aires

ARTICULO 7

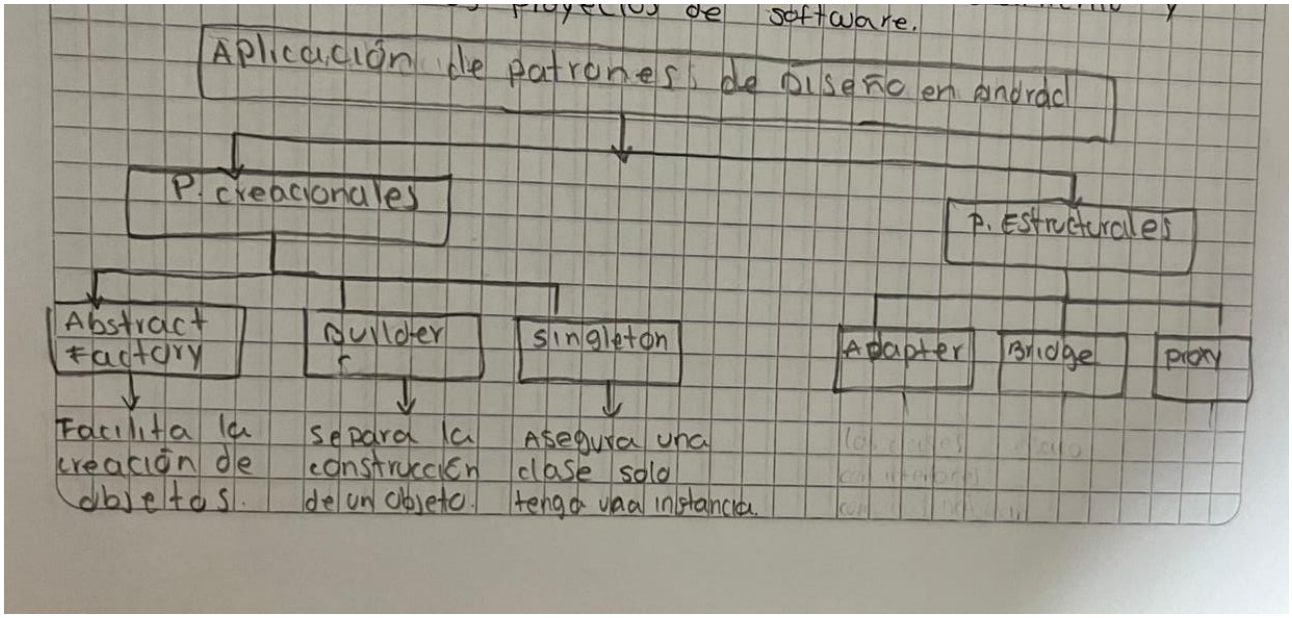
Clasificación de los patrones de diseño idóneos en programación Android

Este artículo examina la aplicación de patrones de diseño en la programación para Android, destacando su importancia para evitar prácticas deficientes como el "código espagueti." Los patrones de diseño, introducidos en la arquitectura y adaptados al software, permiten resolver problemas comunes y hacer el código más legible y eficiente. La investigación identifica y clasifica patrones de diseño idóneos en dos categorías: creacionales y estructurales. Entre los patrones creacionales destacados están Factory Method, Abstract Factory, Builder y Singleton, que facilitan la creación de objetos de forma desacoplada del resto del sistema. En cuanto a los patrones estructurales, se mencionan Adapter, Bridge y Proxy, que optimizan la organización y la flexibilidad del sistema. La implementación de estos patrones permite desarrollar aplicaciones móviles más robustas, adaptables y fáciles de mantener, beneficiando tanto la calidad como el rendimiento de las aplicaciones Android.

Reflexión:

Este artículo deja como reflexión que los patrones de diseño son herramientas esenciales para mejorar la calidad del software, especialmente en el desarrollo de aplicaciones móviles. La correcta implementación de estos patrones permite a los desarrolladores producir código más organizado y modular, reduciendo la redundancia y facilitando el mantenimiento de cada aplicación Android. Además, el uso de patrones fomenta una cultura de buenas prácticas entre programadores, lo cual es especialmente valioso para desarrolladores en constante formación. En un entorno tan competitivo como el de la programación móvil, estos conocimientos sobre patrones de diseño pueden marcar una diferencia significativa en la eficiencia, rendimiento y sostenibilidad de los proyectos de software.

Diagrama:



Bibliografía:

Ibarra A, Luis J, Ulises R, Lepe C, Garzón AP. CLASIFICACIÓN DE LOS PATRONES DE DISEÑO IDÓNEOS EN PROGRAMACIÓN ANDROID. ***Revista Digital de Tecnologías Informáticas y Sistemas*****. 2024;3(1). Accessed November 19, 2024.**

https://redtis.org/index.php/Redtis/article/view/33

ARTICULO 8

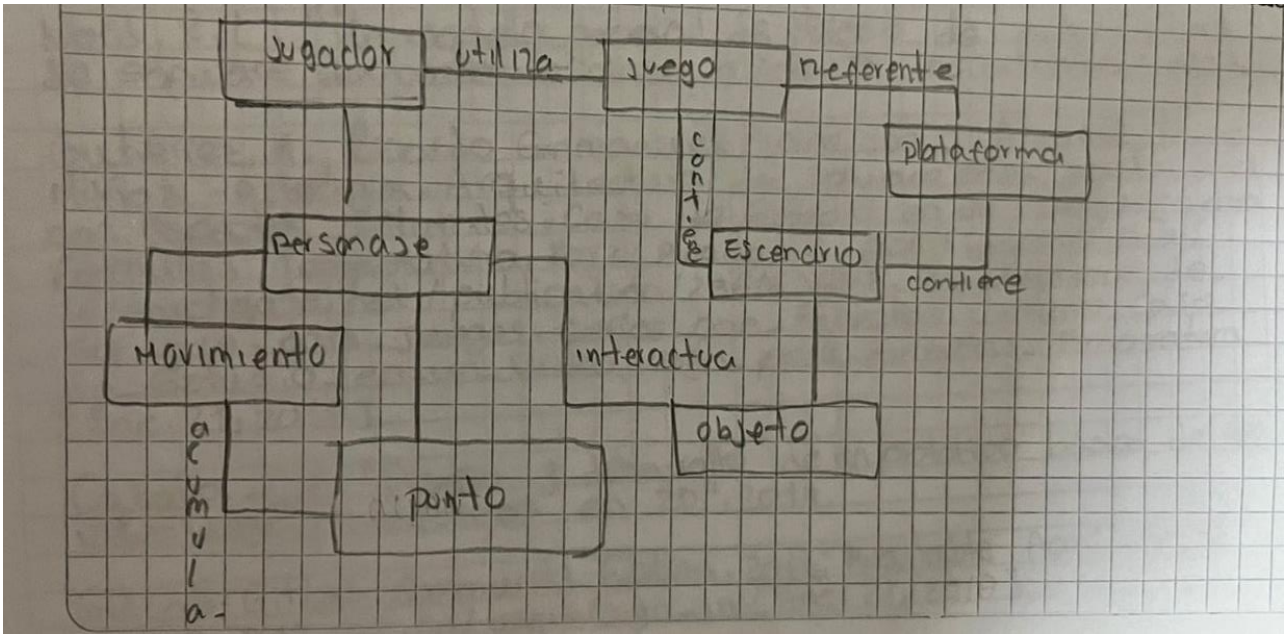
Arquitectura de software para el desarrollo de videojuegos sobre el motor de juego Unity 3D.

Este artículo describe una arquitectura de software diseñada para el desarrollo de videojuegos en el motor Unity 3D, con el objetivo de mejorar la estructura, organización y calidad del código en la creación de estos productos. La propuesta utiliza una arquitectura en capas combinada con componentes, permitiendo una organización modular y flexible. Se detallan tres capas principales: la capa principal del juego, que incluye el controlador del juego y la administración de sonido y datos; la capa de personajes, donde se gestionan los personajes jugables y no jugables; y la capa de interacción con el mundo, que contiene los escenarios y objetos interactivos. Además, la arquitectura incorpora patrones de diseño, como Singleton y Event Handler, para mejorar la eficiencia y control en el manejo de eventos. La validación se realiza mediante el método ATAM, que evalúa atributos de calidad como la mantenibilidad y la extensibilidad, logrando un modelo adecuado para videojuegos multiplataforma. Esta arquitectura proporciona una base robusta y reutilizable para el desarrollo de videojuegos en Unity, facilitando el trabajo colaborativo y la continuidad de proyectos a largo plazo.

Reflexión:

Este artículo deja como reflexión que una arquitectura de software bien diseñada es esencial para el desarrollo eficiente de videojuegos en motores como Unity 3D. La estructura modular y en capas permite que los desarrolladores integren nuevas funcionalidades y realicen mejoras sin afectar la estabilidad del sistema. Además, el uso de patrones de diseño fomenta una mayor organización y control en el flujo de eventos, algo crucial en proyectos de entretenimiento interactivo donde la jugabilidad y la experiencia del usuario son prioritarias. Esta arquitectura no solo facilita el mantenimiento y la escalabilidad de los videojuegos, sino que también asegura una base reutilizable, mejorando el tiempo de desarrollo y permitiendo que los desarrolladores se concentren en la creatividad y la innovación en lugar de lidiar con problemas técnicos.

Diagrama:



Bibliografía:

Andy Hernández Paez 1*, Javier Alejandro Domínguez Falcón 2 , Alejandro Andrés Pi Cruz3 1,2,3 Centro de Entornos Interactivos 3D, Vertex, Universidad de las Ciencias Informáticas. Carretera a San Antonio de los Baños, Km 2½, Torrens, La Lisa, La Habana, Cuba

ARTICULO 9

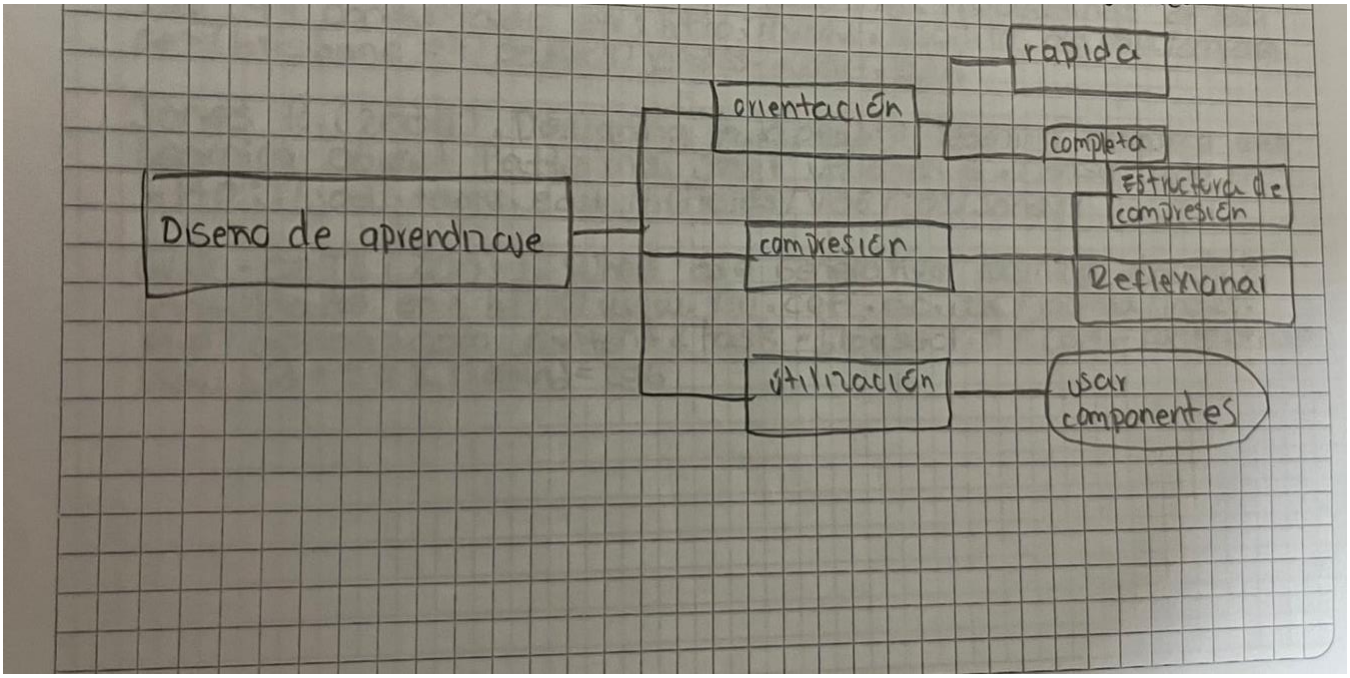
Patrones de Diseño aplicados a la organización de repositorios de objetos de aprendizaje

Este artículo explora cómo los patrones de diseño en la arquitectura de objetos de aprendizaje pueden optimizar la creación y reutilización de recursos educativos digitales. A través de una arquitectura centrada en el Centro de Excelencia (CETL), se propone un modelo estructurado que organiza y facilita el acceso a estos recursos en sistemas de gestión de aprendizaje (LMS). Los repositorios centralizados permiten almacenar y categorizar objetos de aprendizaje, etiquetándolos con metadatos para que sean fácilmente localizables y adaptables a diferentes contextos educativos. Asimismo, se enfatiza en los Objetos de Aprendizaje Generativos (GLO), que brindan una experiencia educativa interactiva y ajustable a las necesidades individuales de docentes y estudiantes. Este enfoque arquitectónico no solo mejora la eficiencia en la distribución de contenidos, sino que también promueve una educación digital inclusiva, adaptable y en constante actualización, adecuada para las exigencias del entorno educativo actual.

Reflexión:

Este artículo deja como reflexión que la arquitectura de objetos de aprendizaje no solo es una herramienta de almacenamiento y distribución de contenidos, sino una estrategia para personalizar y enriquecer el proceso educativo. Al establecer un modelo estructurado y adaptativo, se promueve una educación que responde a las necesidades específicas de los usuarios, fomentando una mayor interacción y compromiso de los estudiantes. Además, el uso de metadatos y categorías en los repositorios facilita el acceso a contenidos pertinentes, ahorrando tiempo y recursos. Este enfoque arquitectónico subraya la importancia de la flexibilidad y la actualización continua de los recursos educativos digitales, permitiendo que la educación en el entorno digital evolucione de manera inclusiva y accesible, adaptándose a los cambios tecnológicos y pedagógicos actuales.

Diagrama:



Bibliografía:

Vista de Patrones de Diseño aplicados a la organización de repositorios de objetos de aprendizaje. Revistas.um.es. Published 2024. Accessed November 19, 2024. https://revistas.um.es/red/article/view/89331/86361

ARTICULO 10

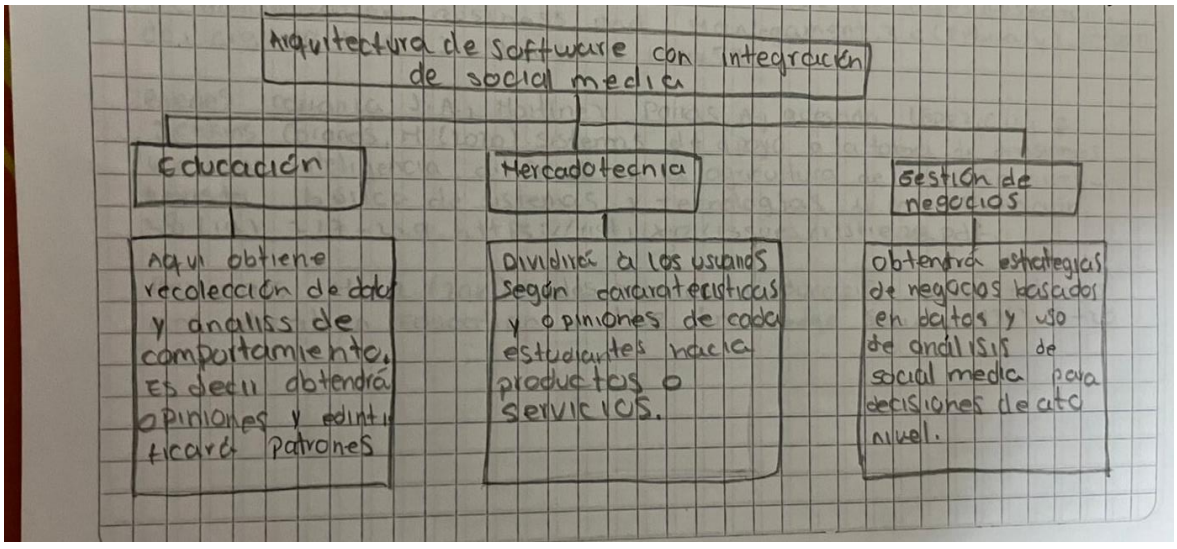
Identificación de áreas de aplicación de arquitecturas de software basadas en modelos, técnicas y herramientas de social media.

El artículo examina cómo las arquitecturas de software pueden aprovechar las técnicas y herramientas de social media para la recolección y análisis de datos generados en redes sociales. Se realiza un mapeo sistemático de literatura para identificar enfoques de aplicación en áreas como la educación, mercadotecnia, gestión de negocios e investigación. El análisis destaca el valor de las arquitecturas que incorporan modelos de análisis de social media para entender comportamientos, opiniones y emociones de los usuarios. Este enfoque permite a las organizaciones ajustar sus estrategias según la dinámica en redes sociales, maximizando la interacción y mejorando la toma de decisiones. También se identifican retos significativos en términos de escalabilidad, interoperabilidad y seguridad, que deben abordarse para lograr una integración eficaz de social media en sistemas de software.

Reflexión:

Este artículo deja como reflexión que la integración de social media en arquitecturas de software representa una evolución necesaria para la adaptación a un entorno digital en constante cambio para un nuevo mundo . La capacidad de analizar grandes volúmenes de datos provenientes de redes sociales permite a las organizaciones estar más conectadas con su audiencia y adaptar sus estrategias de comunicación y negocio de manera precisa. Sin embargo, esta integración plantea también desafíos técnicos y éticos, como asegurar la privacidad y la seguridad de los datos. Este contexto refleja la importancia de desarrollar frameworks robustos y éticos en el ámbito del software, que no solo aprovechen el potencial de social media, sino que también respeten los derechos y la seguridad de los usuarios en plataformas digitales.

Diagrama:



Bibliografía:

Velazquez-Solis PE, Flores-Rios BL, Ibarra-Esquer JE, et al. Identificación de áreas de aplicación de arquitecturas de software basadas en modelos, técnicas y herramientas de social media. *RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação*. 2021;(42):12-29. doi:https://doi.org/10.17013/risti.42.12-29

ARTICULO 11

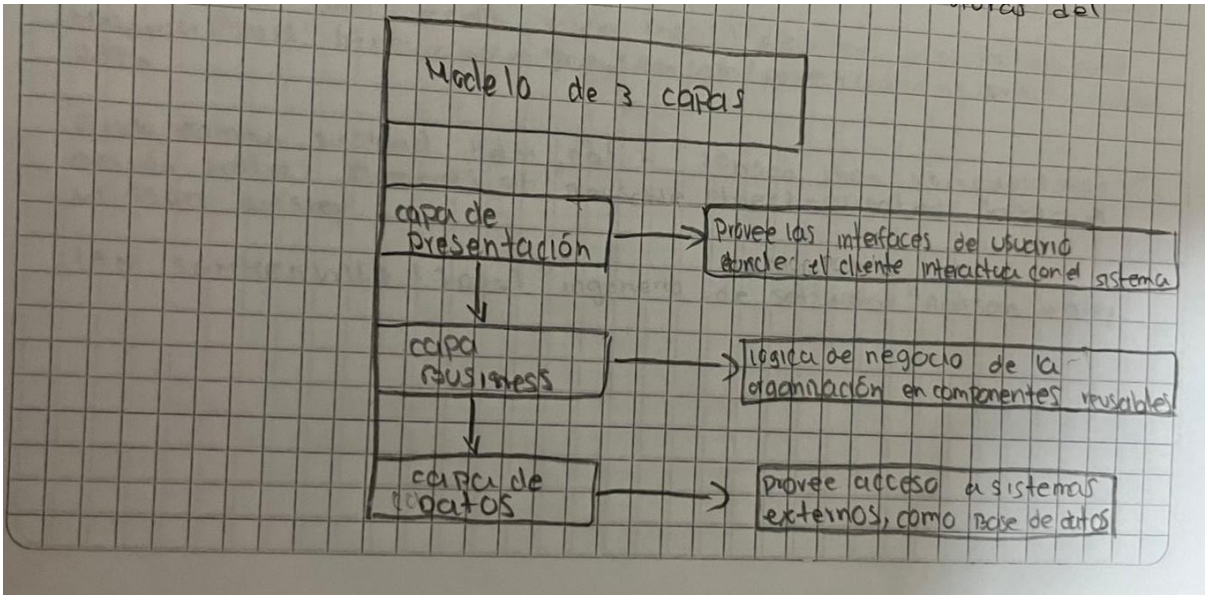
Desarrollo de sistemas de software con patrones de diseño orientado a objetos.

El artículo explora cómo implementar patrones de diseño orientados a objetos para desarrollar una arquitectura de software en la Facultad de Ingeniería Industrial, centrada en la creación de una intranet llamada "Intranet Industrial." Su objetivo es evaluar los beneficios económicos de usar patrones de diseño frente a enfoques tradicionales. En la investigación, se destacan patrones como el de tres capas, informador y sensor, que optimizan la estructura del sistema y reducen los costos de mantenimiento. Se emplea la metodología COCOMO para comparar costos y efectividad en el desarrollo de software con y sin patrones, subrayando la importancia del modularidad y la reutilización de componentes. La implementación de esta arquitectura aporta eficiencia y permite a la intranet adaptarse a futuras modificaciones y escalabilidad. La investigación concluye que los patrones de diseño arquitectónico son una base sólida para el desarrollo de sistemas en proyectos complejos de software, destacando su aplicabilidad y beneficios.

Reflexión:

Este artículo deja como reflexión que adoptar patrones de diseño en la arquitectura de software es crucial para lograr sistemas eficientes, escalables y de fácil mantenimiento. La "Intranet Industrial" muestra cómo una arquitectura bien organizada reduce costos, facilita el mantenimiento y mejora la claridad del código. Los patrones de diseño como el de tres capas y el sensor permiten construir sistemas modulares y adaptables a largo plazo, lo cual es esencial en entornos de constante cambio. La reutilización y modularidad logradas con patrones orientados a objetos no solo optimizan el proceso de desarrollo, sino que también brindan a los equipos una guía para enfrentar la complejidad de los sistemas en crecimiento. Así, los patrones arquitectónicos ofrecen una estructura sostenible para crear aplicaciones robustas y flexibles que responden a las demandas futuras del mercado y de los usuarios.

Diagrama:



Bibliografía:

CHRISTOPHER ALEXANDER (1977) "A Pattern Language" (Oxford University Press 1977).

CHRISTOPHER ALEXANDER (1977) "EL Modo Intemporal De Construir" (Oxford University Press 1985).

"The Software Patterns Criteria (Proposed Definitions for Evaluating Software Pattern Quality)" Version 10.2 June 2, 1998 http://www.antipatterns.com/whatisapattern/.

ARTICULO 12

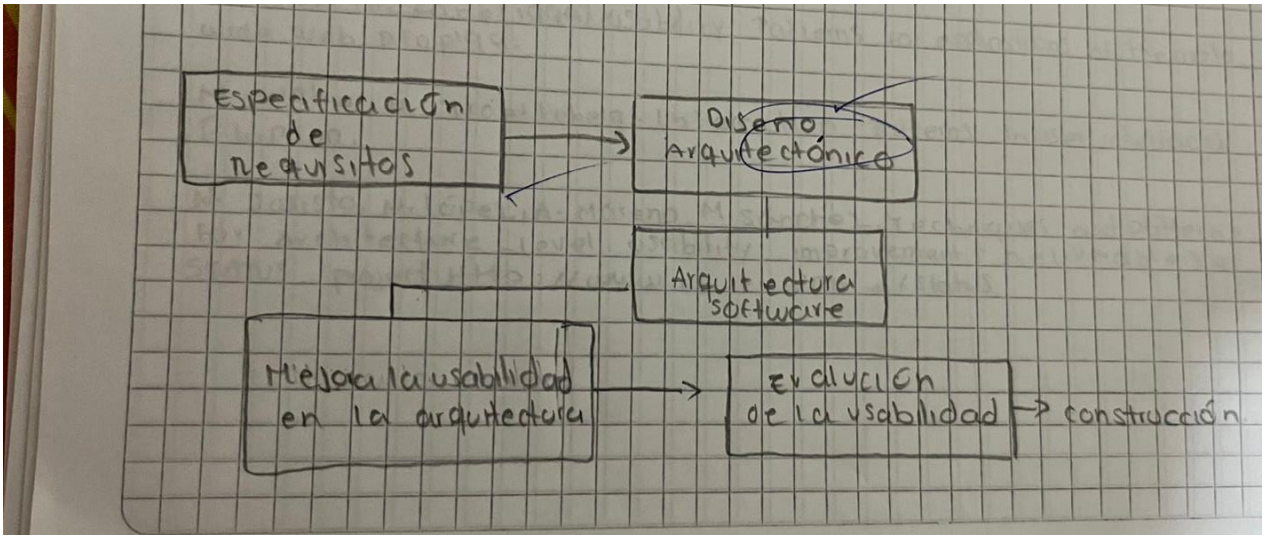
Patrones de Usabilidad: Mejora de la Usabilidad del Software desde el momento de Arquitectónico.

El artículo explora cómo mejorar la usabilidad en el desarrollo de software a través de la arquitectura, enfatizando el proyecto STATUS que aboga por incorporar atributos de usabilidad desde el inicio del diseño. En lugar de ajustar la usabilidad al final del desarrollo, STATUS propone patrones arquitectónicos específicos, como feedback y guía de usuario, que integran propiedades de usabilidad desde la fase arquitectónica. Este enfoque asegura que los sistemas construidos puedan facilitar una experiencia de usuario más fluida, considerando atributos clave como la satisfacción, el aprendizaje y la eficiencia. El proyecto también destaca cómo los patrones de usabilidad, tales como el feedbacker o undoer, no solo guían al usuario, sino que mejoran la interacción en aplicaciones complejas. Estos patrones se aplican en arquitecturas modulares y en soluciones de diseño reutilizables, beneficiando tanto la interacción como la satisfacción del usuario final.

Reflexión:

Este artículo deja como reflexión la importancia de diseñar software que considere la usabilidad desde la fase inicial. Incorporar atributos de usabilidad en la arquitectura garantiza sistemas más satisfactorios y eficientes para el usuario, evitando problemas y ajustes tardíos. STATUS propone un enfoque innovador que permite crear aplicaciones funcionales y amigables mediante patrones específicos, promoviendo prácticas de diseño centradas en el usuario. Este enfoque plantea que la estructura y la usabilidad del software están interconectadas y que al integrar usabilidad desde el diseño se mejora la experiencia final del usuario. Además, se resalta que diseñar con usabilidad en mente no solo optimiza el rendimiento del sistema, sino que también reduce costos a largo plazo al minimizar la necesidad de refactorizaciones o modificaciones. Por último, esta perspectiva invita a los desarrolladores a considerar a los usuarios como una pieza clave del diseño, priorizando sus necesidades para construir software más adaptable y eficaz.

Diagrama:



Bibliografía:

Patrones de Usabilidad: Mejora de la Usabilidad del Software desde el momento de Arquitectónico Ana M. Moreno Facultad de Informática Universidad Politécnica de Madrid, España M. Isabel Sánchez Facultad de Informática Universidad Carlos III de Madrid, España

ARTICULO 13

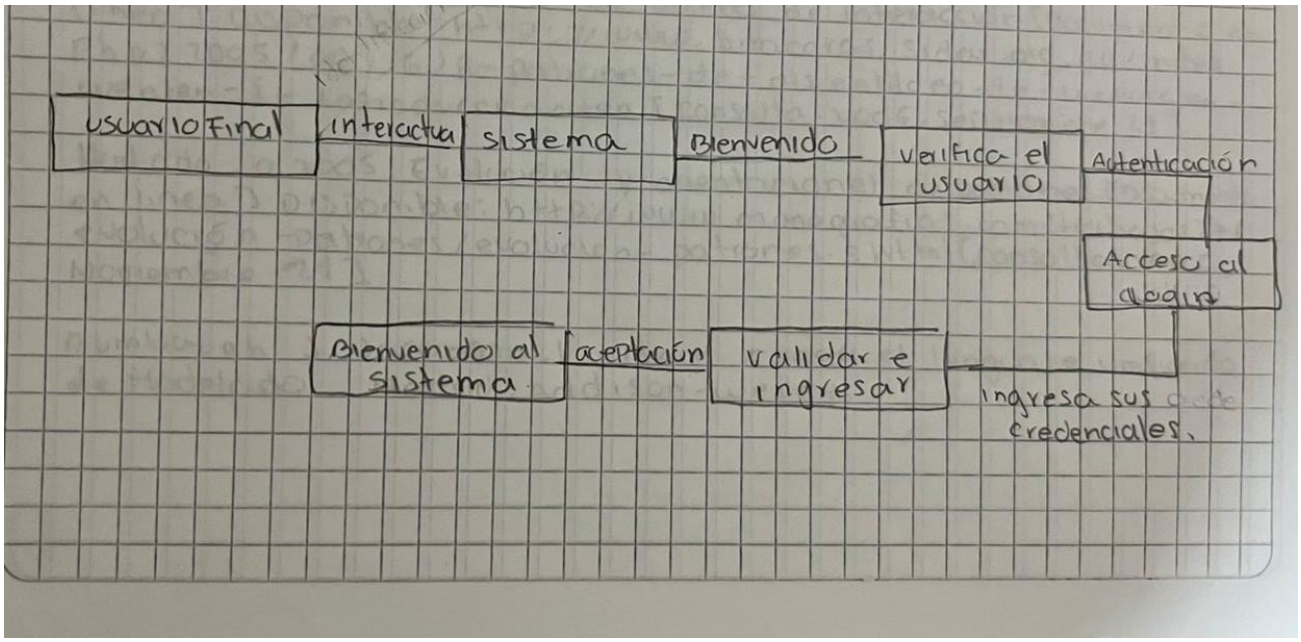
Herramienta De Reúso JavaScript Orientado a Patrones De Interacción.

El artículo presenta la herramienta ReusMe, diseñada para mejorar la reutilización de código JavaScript en el desarrollo de interfaces de usuario basadas en patrones de interacción. ReusMe permite a los desarrolladores de aplicaciones web acceder y personalizar código reusable, facilitando la creación de interfaces usables y optimizadas sin necesidad de comenzar desde cero. Basándose en metodologías como el UML y el Proceso Unificado de Desarrollo de Software (PUD), la herramienta integra patrones de interacción para crear componentes web que se ajusten a las necesidades del usuario final. La reutilización de patrones de interacción ayuda a mejorar la eficiencia en el diseño de interfaces, reduciendo tiempos y costos, y asegurando una experiencia de usuario coherente y de alta calidad. ReusMe representa una alternativa a las guías de diseño tradicionales al proporcionar soluciones listas para ser implementadas, aumentando la consistencia y calidad de los proyectos web.

Reflexión:

Este artículo deja como reflexión que la reutilización de código a través de patrones de interacción es clave para la eficiencia y calidad en el desarrollo de interfaces de usuario en aplicaciones web. ReusMe ofrece a los diseñadores web una herramienta práctica y accesible, que no solo ahorra tiempo, sino que también mejora la experiencia del usuario mediante la implementación de patrones probados y efectivos. La idea de "no reinventar la rueda" es particularmente relevante en este contexto, ya que permite a los desarrolladores concentrarse en otros aspectos importantes de la aplicación en lugar de crear desde cero. Este enfoque de reutilización destaca la importancia de contar con herramientas que estandaricen el diseño y mantengan la coherencia visual y funcional en las interfaces, mejorando así la usabilidad y satisfacción del usuario final.

Diagrama:



Bibliografía:

BENIGNI, GLADYS; ANTONELLI, OCTAVIO; VÁSQUEZ, YOVANNY

HERRAMIENTA PARA REUSO DE CÓDIGO JAVASCRIPT ORIENTADO A

PATRONES DE INTERACCIÓN SABER. Revista Multidisciplinaria del Consejo de

Investigación de la Universidad de Oriente, vol. 21, núm. 1, enero-abril, 2009, pp. 60-69

Universidad de Oriente Cumaná, Venezuela

ARTICULLO 14

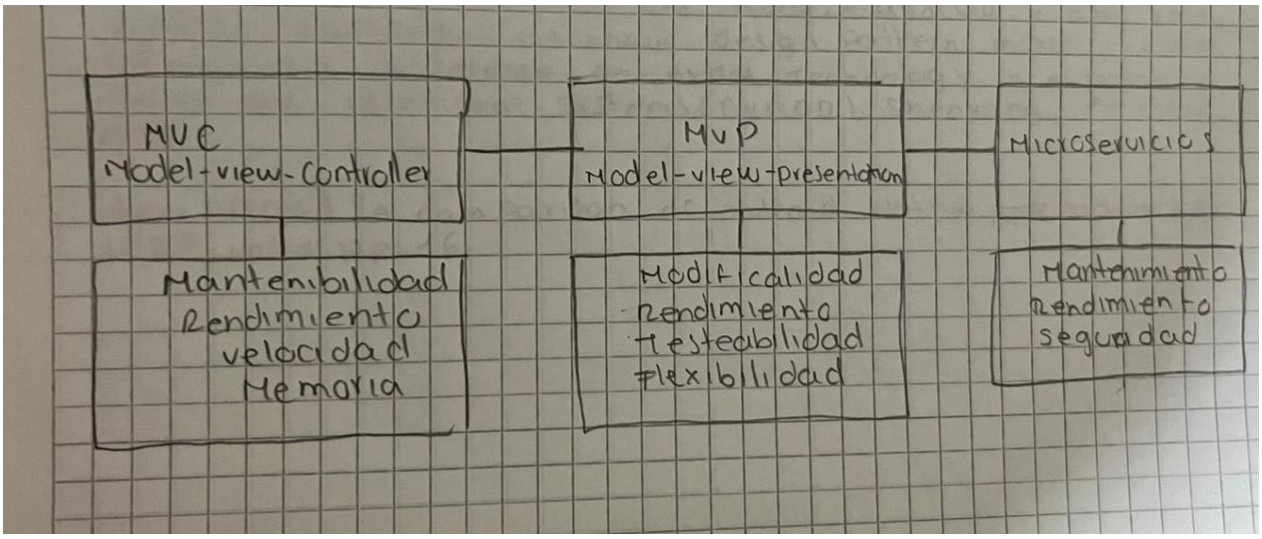
Monolitos vs. Microservicios en Arquitectura de Software: Perspectivas para un Desarrollo Eficiente.

Este artículo analiza las diferencias entre las arquitecturas de software monolíticas y de microservicios, explorando sus ventajas y limitaciones en el desarrollo moderno. Las arquitecturas monolíticas concentran toda la funcionalidad en un único sistema, lo que simplifica el despliegue y la gestión en proyectos pequeños, pero puede dificultar la escalabilidad en aplicaciones más complejas. Por otro lado, los microservicios dividen la aplicación en componentes independientes que interactúan mediante interfaces, permitiendo una mayor flexibilidad y escalabilidad en entornos dinámicos. Sin embargo, los microservicios presentan desafíos en la coordinación y requieren una infraestructura más compleja. Empresas como Amazon y eBay han adoptado microservicios para responder a la demanda del mercado, mientras que otros proyectos, como algunos de Walmart, han utilizado una arquitectura monolítica para reducir costos y simplificar procesos.

Reflexión:

Este artículo deja como reflexión que la elección de una arquitectura debe estar alineada con las necesidades específicas del proyecto y las capacidades del equipo de desarrollo. La arquitectura monolítica puede ser eficaz en aplicaciones pequeñas o con requisitos estables, proporcionando simplicidad y reduciendo la carga operativa. En cambio, los microservicios, aunque más complejos, ofrecen una mayor adaptabilidad para proyectos de gran escala y en constante cambio. Es esencial que los arquitectos de software analicen las características de cada proyecto, anticipando necesidades futuras y considerando tanto los beneficios como los desafíos operativos de cada arquitectura.

Diagrama:



Bibliografía:

Valentín Torassa Colombero, Estelles JP, Gallegos L, Lopez P. Monolitos vs. Microservicios en Arquitectura de Software: Perspectivas para un Desarrollo Eficiente. ***Memorias de las JAIIO*****. 2024;10(5):42-54. Accessed November 19, 2024.**

https://n9.cl/cjp3u

ARTICULO 15

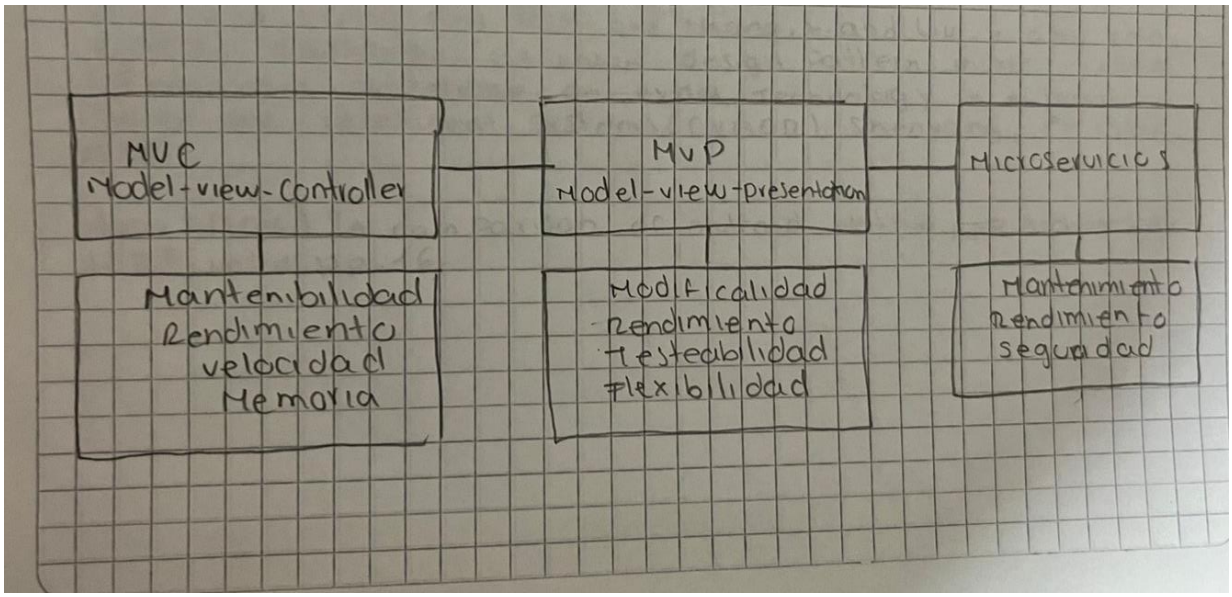
Marco de Trabajo para Seleccionar un Patrón Arquitectónico en el Desarrollo de Software.

Este artículo presenta un marco de trabajo diseñado para seleccionar el patrón arquitectónico más adecuado en el desarrollo de software, considerando factores clave como la calidad, modularidad y escalabilidad. La propuesta identifica patrones como MVC, MVP, microservicios y arquitecturas en la nube, cada uno con características específicas que se adaptan a distintos tipos de aplicaciones. Este marco de trabajo permite a los arquitectos de software y desarrolladores elegir el patrón más idóneo en función de variables como el tipo de aplicación (web, móvil, escritorio) y requerimientos de seguridad, rendimiento y mantenibilidad. La implementación de este marco no solo facilita la selección del patrón adecuado, sino que también optimiza los tiempos y costos, al reducir el riesgo de reprocesos y problemas de escalabilidad. Se destacan las reglas establecidas dentro del marco, que guían a los desarrolladores en la toma de decisiones según las necesidades del proyecto, asegurando que el producto final cumpla con los estándares de calidad esperados.

Reflexión:

Este artículo deja como reflexión la importancia de contar con un marco de trabajo claro para la selección de patrones arquitectónicos, lo cual es fundamental en proyectos de software de alta calidad. Al disponer de una guía que permita a los desarrolladores elegir el patrón más adecuado para su proyecto, se minimizan riesgos de fallos en la estructura del software y se promueve la escalabilidad y sostenibilidad del producto final. Este enfoque muestra que una arquitectura de software bien planificada no solo responde a los requisitos actuales de la aplicación, sino que también se anticipa a futuras demandas de mantenimiento y actualización. Implementar un marco como el propuesto aporta claridad en las decisiones arquitectónicas y fomenta prácticas de desarrollo que se alinean con las mejores prácticas de la industria, asegurando que cada capa del software esté desacoplada y optimizada para su función específica.

Diagrama:



Bibliografía:

Camilo J, Alberto, Kelly G. Marco de Trabajo para Seleccionar un Patrón Arquitectónico en el Desarrollo de Software. *Tdeaeduco*. Published online 2022. doi:https://dspace.tdea.edu.co/handle/tdea/2670

ARTICULO 16

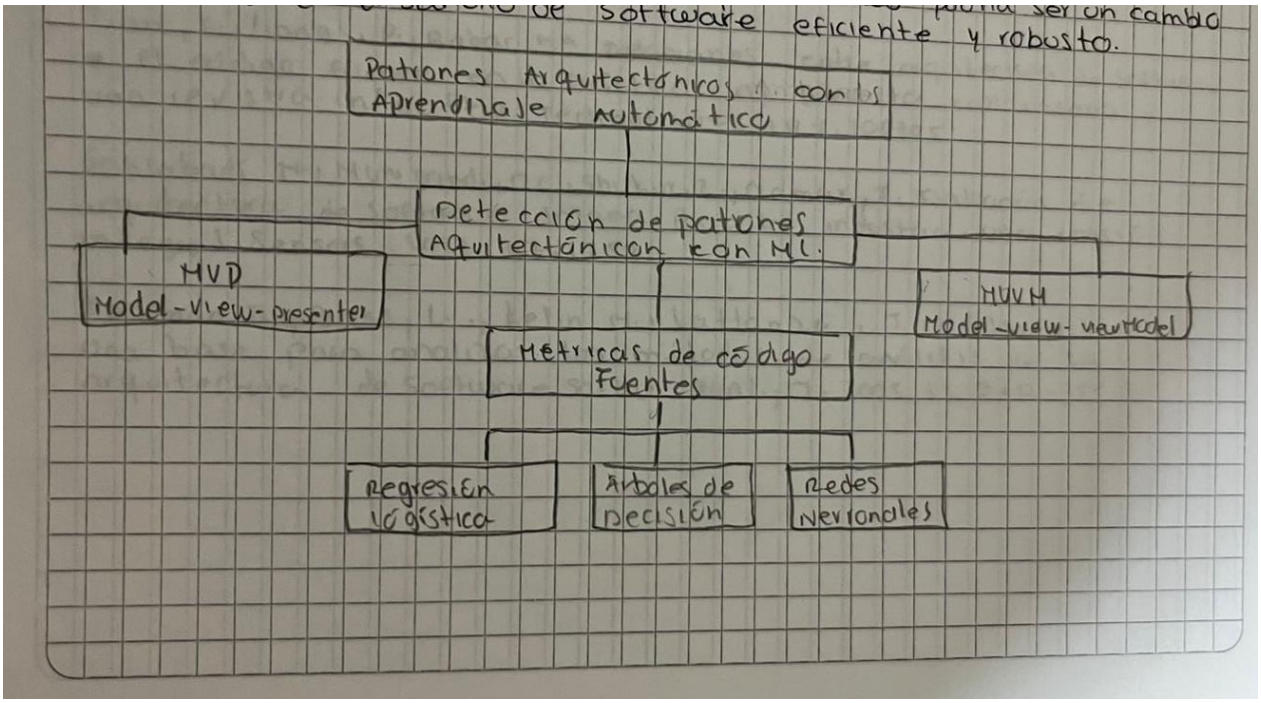
Hacia la predicción de patrones de diseño arquitectónico: un enfoque de aprendizaje automático controladores lógicos programables.

Este artículo presenta un enfoque basado en aprendizaje automático (ML) para predecir patrones de diseño arquitectónico en proyectos de software, específicamente los patrones MVP (Model-View-Presenter) y MVVM (Model-View-ViewModel). La investigación usa métricas de código fuente, extraídas de más de 5,000 proyectos en Java de GitHub, para entrenar nueve modelos ML que alcanzan una precisión del 83%. Los modelos se validan con un k-fold (k = 5), y el enfoque demuestra su efectividad en identificar patrones arquitectónicos con un conjunto mínimo de métricas, mejorando resultados previos en la detección de patrones. Además, el artículo destaca cómo el aprendizaje automático puede ayudar en la detección de patrones y la relación entre las métricas de código y la arquitectura de software, optimizando la consistencia entre el código y la arquitectura planificada. Los autores también abordan las limitaciones de los datos y los retos de validación en ML, destacando la importancia de las métricas clave en la elección de patrones arquitectónicos y en la evaluación de la degradación de la arquitectura.

Reflexión:

Este artículo deja como reflexión que el aprendizaje automático es una herramienta poderosa para optimizar la toma de decisiones en la arquitectura de software, permitiendo detectar patrones arquitectónicos con precisión y mejorando la consistencia entre la arquitectura planificada y la implementación. La investigación abre nuevas perspectivas para la automatización en ingeniería de software, especialmente en entornos donde la elección de patrones afecta directamente la mantenibilidad y escalabilidad del proyecto. Esto sugiere que integrar técnicas de ML en el proceso de diseño arquitectónico podría ser un cambio clave para el desarrollo de software eficiente y robusto. La creación de un dataset amplio y las métricas destacadas ofrecen una base sólida para futuras investigaciones y aplicaciones prácticas en el ámbito de la ingeniería de software.

Diagrama:



Bibliografía:

Sirojiddin Komolov, Gcinizwe Dlamini, Swati Megha, Mazzara M. Towards Predicting Architectural Design Patterns: A Machine Learning Approach. *Computers*. 2022;11(10):151-151. doi:https://doi.org/10.3390/computers11100151

ARTICULO 17

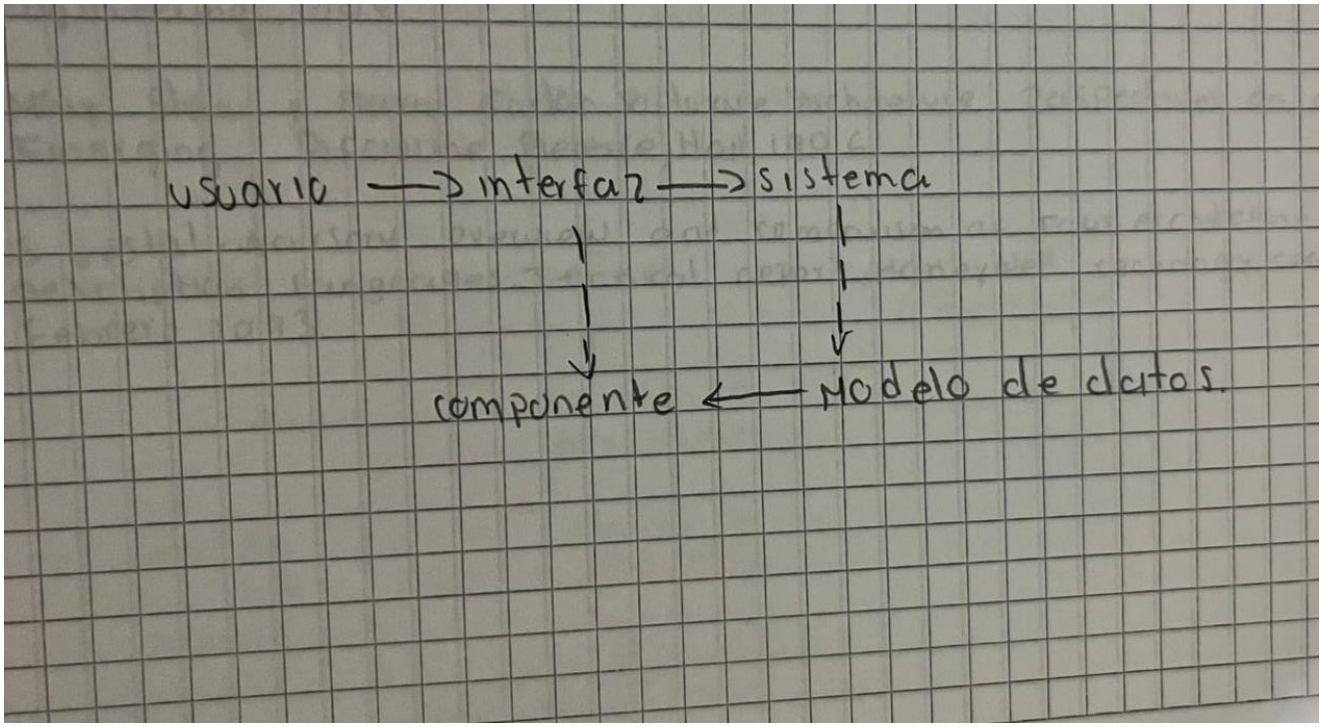
Representación de la arquitectura de software usando UML

El artículo describe la evolución de la arquitectura de software y la necesidad de representaciones precisas que permitan una comunicación clara entre todos los involucrados en un proyecto. Explica cómo los lenguajes de descripción de arquitecturas (ADL) actuales suelen ser complejos y específicos para ciertos sistemas, lo que dificulta su adopción general en la industria. En este contexto, el UML se presenta como una alternativa flexible y conocida que permite modelar diferentes aspectos de un sistema mediante diagramas diversos como los de casos de uso, clases, componentes, entre otros. Estos diagramas ayudan a capturar las estructuras esenciales de un sistema, facilitando su análisis y la toma de decisiones durante el desarrollo y mantenimiento. Además, UML incluye mecanismos de extensión como estereotipos y restricciones, que permiten personalizar la representación para adaptarse a las características únicas de cada sistema, mejorando la precisión y utilidad del modelo arquitectónico.

Reflexión:

Este artículo deja como reflexión la importancia de contar con herramientas de modelado que sean tanto precisas como comprensibles para todos los participantes de un proyecto de software. Utilizar un lenguaje estándar como UML en la representación de la arquitectura permite que el conocimiento sobre el sistema sea accesible y reutilizable en todas las etapas de su ciclo de vida. Esto no solo mejora la comunicación entre desarrolladores y otros interesados, sino que también facilita el mantenimiento y la evolución del software. La posibilidad de extender UML mediante estereotipos y restricciones muestra cómo la flexibilidad es clave en herramientas de modelado, permitiendo adaptarse a diversos estilos arquitectónicos sin perder la estandarización, lo cual es esencial para el éxito en la ingeniería de software a largo plazo.

Diagrama:



Bbliografía:

View of UML-based Scheme for Software Architecture Representations. Icesi.edu.co. Published 2024. Accessed November 19, 2024.

https://n9.cl/jo7i9

ARTICULO 18

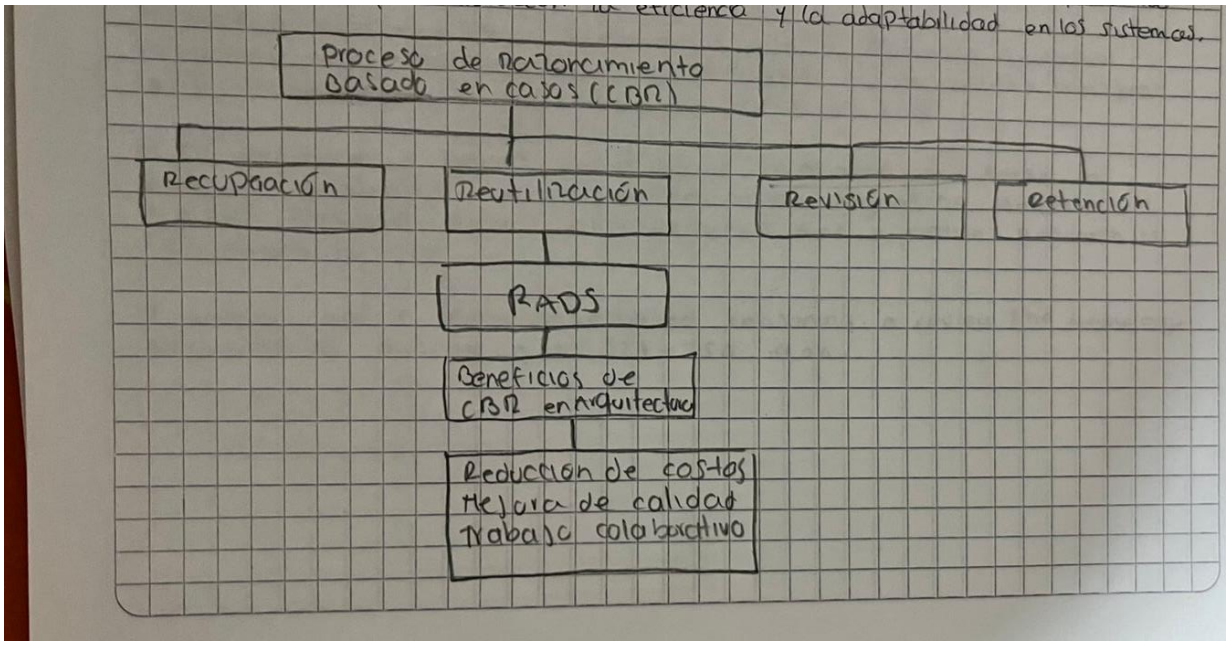
Revisión de elementos conceptuales para la representación de las arquitecturas de referencias de software

Este artículo explora los conceptos clave para la representación de arquitecturas de referencia de software (ARS). La arquitectura de software define la estructura y organización de un sistema, estableciendo componentes y relaciones sin profundizar en detalles internos. Existen diversas formas de describir las ARS, destacando el uso de lenguajes de descripción arquitectónica (ADL) que permiten modelar, analizar y simular arquitecturas. Los ADL como UniCon, Wright y Darwin representan componentes y conectores, ofreciendo enfoques para arquitecturas dinámicas o formales, aunque presentan limitaciones en la reutilización y adaptación a cambios. Además, se mencionan marcos de referencia como RM-ODP y TOGAF, que estructuran la arquitectura en vistas, facilitando la comunicación y estandarización. El artículo concluye con una propuesta para mejorar la representación del conocimiento arquitectónico, apoyando decisiones y razonamientos arquitectónicos.

Reflexión:

Este artículo deja como reflexión la importancia de estandarizar y formalizar la arquitectura de software, no solo para mejorar la comunicación entre equipos, sino también para asegurar la integridad y consistencia de los sistemas a lo largo de su ciclo de vida. A medida que los sistemas de software se vuelven más complejos y requieren interoperabilidad en entornos diversos, contar con modelos y marcos de referencia bien definidos facilita la adaptación y evolución de los sistemas sin comprometer su calidad. El trabajo resalta la necesidad de herramientas y lenguajes específicos que permitan la representación precisa de arquitecturas, fomentando la reutilización y adaptación a nuevos requerimientos, un enfoque que podría revolucionar la manera en que los arquitectos de software abordan el diseño y desarrollo de sistemas.

Diagrama:



Bibliografía:

Angel M, Martínez, Nemury Silega, Yarisbel O, Angel M, Martínez, Nemury Silega, Yarisbel O. Revisión de elementos conceptuales para la representación de las arquitecturas de referencias de software. *Revista Cubana de Ciencias Informáticas*. 2019;13(1):143-157. Accessed November 19, 2024. http://scielo.sld.cu/scielo.php?pid=S2227-18992019000100143&script=sci_arttext

ARTICULO 19

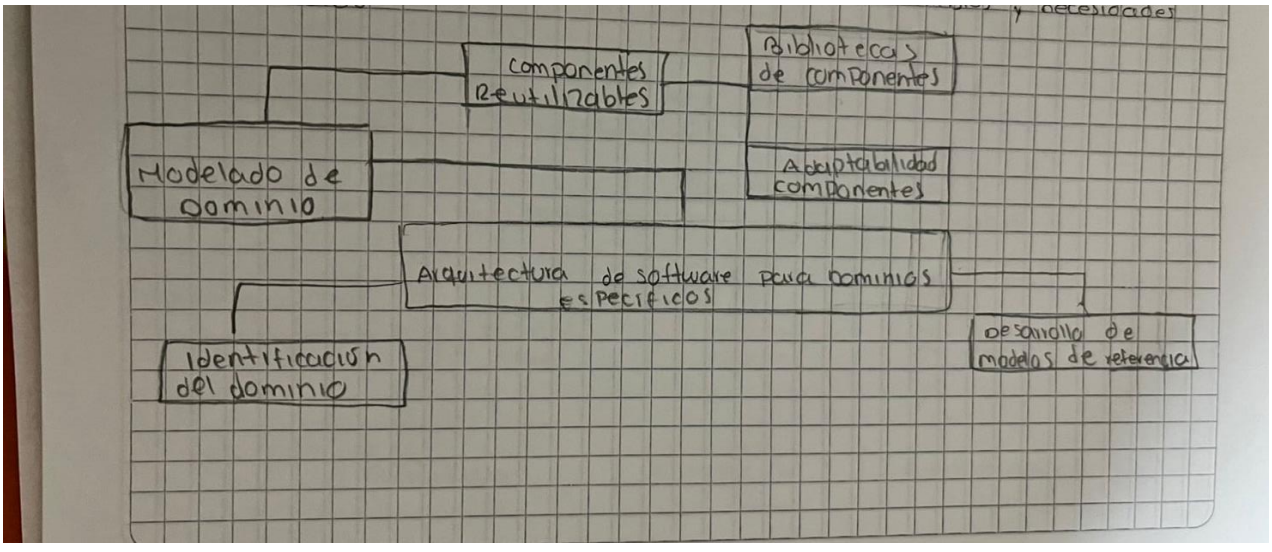
Representación y razonamiento sobre las decisiones de diseño de arquitectura de software.

Este artículo explora cómo la arquitectura de software permite establecer una estructura organizada en el desarrollo de sistemas, facilitando la comunicación, el análisis y el cumplimiento de requisitos de calidad. Las arquitecturas bien diseñadas actúan como planos de los sistemas, ayudando a prever cambios y asegurando su integridad. La técnica de Razonamiento Basado en Casos (CBR) es presentada como una metodología que permite a los arquitectos de software reutilizar conocimientos y experiencias pasadas para enfrentar nuevos desafíos de diseño, ofreciendo soluciones inspiradas en casos similares previamente documentados. El proceso de CBR incluye actividades iterativas como la recuperación, reutilización, revisión y retención de casos arquitectónicos, que permiten adaptar y mejorar las decisiones de diseño. Este enfoque reduce costos, incrementa la calidad del producto y facilita el trabajo colaborativo entre arquitectos a través de una memoria de casos compartida. Además, se detalla el desarrollo de una herramienta de software, RADS, que implementa CBR para apoyar a los arquitectos en la toma de decisiones informadas durante el diseño arquitectónico de software.

Reflexión:

Este artículo deja como reflexión la importancia de contar con una base de conocimiento estructurada y accesible para los arquitectos de software, facilitando la reutilización de experiencias exitosas y la mejora continua en el diseño arquitectónico. La técnica de Razonamiento Basado en Casos no solo reduce la carga de trabajo de los arquitectos al ofrecer soluciones probadas, sino que también promueve un enfoque colaborativo donde la memoria compartida de casos permite un aprendizaje colectivo y una mayor coherencia en los diseños. Esta propuesta subraya la necesidad de herramientas que permitan capturar y formalizar el conocimiento arquitectónico de una manera reutilizable y accesible, optimizando la eficiencia y calidad de los sistemas de software. La incorporación de métodos como CBR en el proceso de diseño arquitectónico evidencia el valor de integrar prácticas de inteligencia artificial para mejorar la toma de decisiones y hacer frente a los crecientes retos de la ingeniería de software moderna.

Diagrama:



Bibliografía:

María A, Carignano C. *Representación Y Razonamiento Sobre Las Decisiones de Diseño de Arquitectura de Software*. Accessed November 19, 2024.

https://n9.cl/umj6h

ARTICULO 20

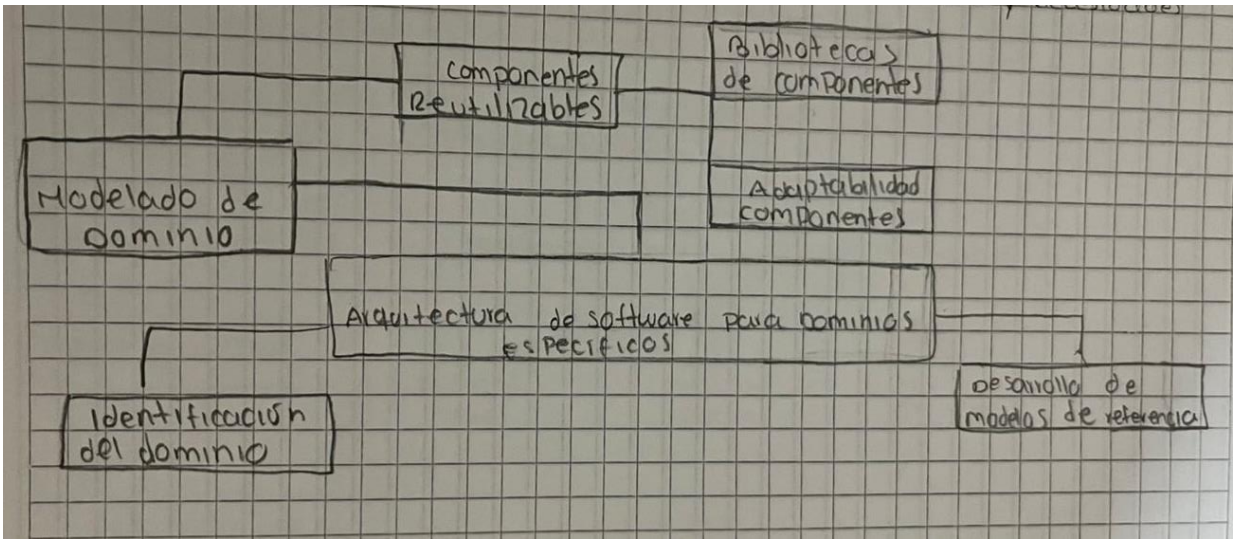
Arquitectura de Software para Dominio Específico (DSSA) : Su Aplicación en la Reutilización de Software

Este artículo presenta las Arquitecturas de Software para Dominios Específicos (DSSA) como una herramienta valiosa para gestionar la reutilización de software en contextos específicos. Las DSSA capturan las similitudes entre sistemas dentro de un mismo dominio, permitiendo que componentes de software predefinidos sean ensamblados y reutilizados en nuevas aplicaciones, optimizando así el proceso de desarrollo. Mediante un enfoque de análisis y modelado de dominios, se identifican elementos comunes y sus relaciones, definiendo una arquitectura genérica aplicable en múltiples contextos dentro del mismo ámbito. Este proceso de ingeniería de dominios permite reducir los costos y mejorar la eficiencia en el desarrollo de aplicaciones, al proporcionar una "caja negra" reutilizable que cumple con requisitos específicos. Asimismo, se describe un marco de trabajo en cinco etapas para construir una DSSA, desde la identificación del dominio hasta el desarrollo de modelos de referencia, integrando necesidades del cliente con restricciones de implementación. El uso de DSSA facilita la producción de software adaptable y contribuye al desarrollo de bibliotecas de componentes reutilizables.

Reflexión:

Este artículo deja como reflexión que contar con arquitecturas específicas para dominios permite maximizar la reutilización y optimización en el desarrollo de software, especialmente en áreas con necesidades comunes. Las DSSA representan una solución eficaz para capturar el conocimiento arquitectónico acumulado en un dominio y aplicarlo de forma efectiva a través de componentes reutilizables, reduciendo así la necesidad de desarrollar desde cero y minimizando los errores. Además, el enfoque de DSSA fomenta la innovación, ya que permite a los ingenieros centrar sus esfuerzos en mejorar y adaptar los componentes existentes en lugar de reinventarlos. Esta metodología también fortalece la colaboración y la transferencia de conocimientos entre proyectos similares, impulsando el avance en la creación de sistemas más robustos y adaptables a futuras exigencias. En un mundo donde la eficiencia y la calidad son cruciales, las DSSA representan una herramienta estratégica que mejora el ciclo de vida del software y permite adaptarse a los cambios tecnológicos.

Diagrama:



Bibliografía:

Oviedo S, Araya D, Zapata SG, Cáceres A. Arquitectura de software para dominio específico (DSSA): su aplicación en la reutilización de software. *Unlpeduar*. Published online May 1999. doi:http://sedici.unlp.edu.ar/handle/10915/22260

ARTICULO 21

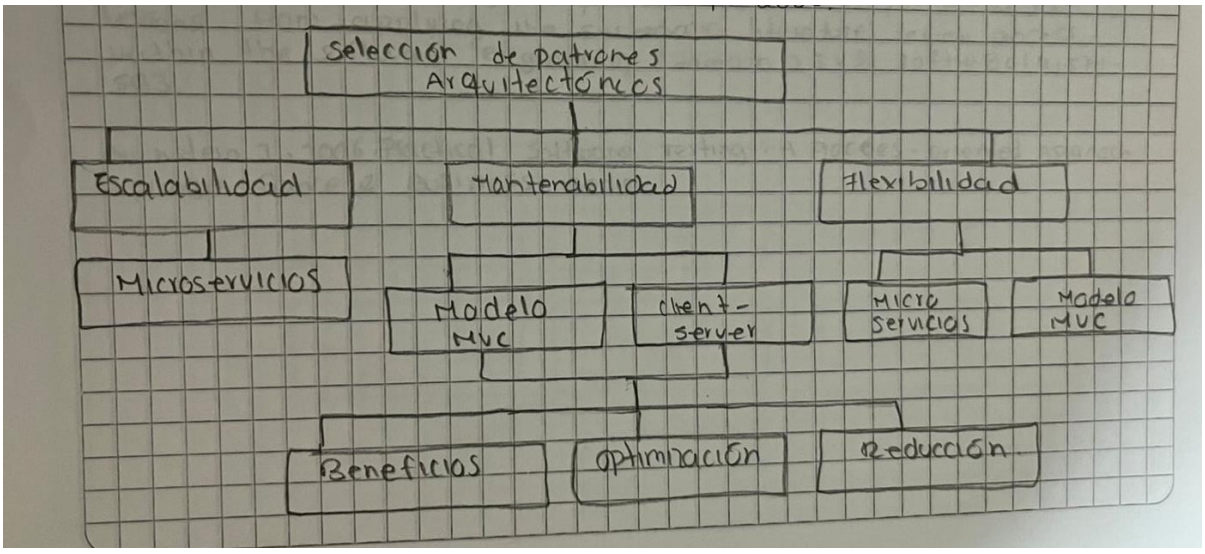
Captura de conocimientos de arquitectura de software para un diseño basado en patrones.

El artículo aborda la importancia de los patrones de arquitectura de software y su papel en el diseño de sistemas eficaces. Los arquitectos de software enfrentan desafíos al seleccionar patrones arquitectónicos debido a que el conocimiento sobre ellos está disperso en diversas fuentes. El estudio tiene como objetivo crear un modelo de toma de decisiones que apoye a los arquitectos, vinculando patrones de software con atributos de calidad específicos. A través de una revisión sistemática de la literatura (SLR), los autores identifican 29 patrones y analizan sus impactos en 40 atributos de calidad, permitiendo a los arquitectos tomar decisiones informadas basadas en investigaciones previas y evaluaciones prácticas. El estudio concluye que proporcionar este conocimiento consolidado ayuda a los arquitectos a seleccionar patrones de manera más eficiente, abordando tanto los requisitos funcionales como los de calidad. Además, se exploran tendencias en el uso de patrones y combinaciones de patrones, sugiriendo la importancia de la adaptabilidad en la arquitectura para cumplir con las necesidades tecnológicas en evolución.

Reflexión:

Este artículo resalta el papel crucial del conocimiento organizado y exhaustivo en el diseño arquitectónico. Al consolidar la información dispersa sobre patrones arquitectónicos, los arquitectos pueden optimizar sus procesos de toma de decisiones, asegurando que la estructura del software cumpla tanto con los requisitos funcionales como con los de calidad. La investigación enfatiza que las decisiones arquitectónicas adaptables y basadas en el conocimiento no solo ahorran tiempo, sino que también mejoran la calidad y la mantenibilidad general de los sistemas de software. Además, este enfoque fomenta una visión más estratégica, donde las elecciones arquitectónicas se alinean con la evolución tecnológica y los requisitos del mercado. Incorporar prácticas que documentan y evalúan los patrones con un enfoque sistemático no solo beneficia a los proyectos actuales, sino que establece una base sólida para futuros desarrollos en la ingeniería de software, promoviendo un diseño más eficiente, sostenible y accesible para arquitectos y equipos de desarrollo.

Diagrama:



Bibliografía:

Siamak Farshidi, Jansen S, Martijn J. Capturing software architecture knowledge for pattern-driven design. *Journal of Systems and Software*. 2020;169:110714-110714. doi:https://doi.org/10.1016/j.jss.2020.110714

ARTICULO 22

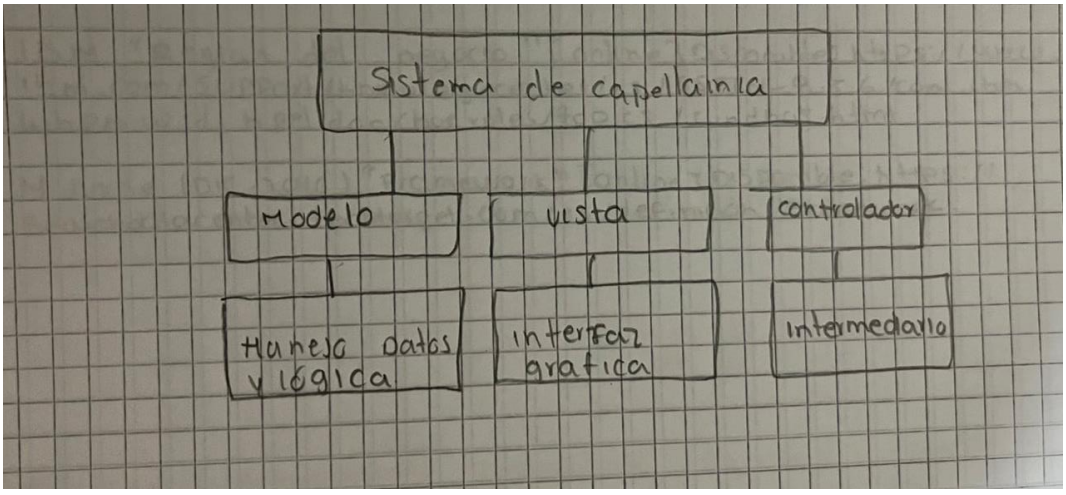
Implementación del patrón arquitectónico MVC en aplicaciones webpara la arquitectura del software del sistema de capellanía de la UM.

El artículo explora la implementación del patrón arquitectónico Modelo-Vista-Controlador (MVC) en una aplicación web para el sistema de capellanía de la Universidad de Montemorelos, con el propósito de mejorar la gestión y accesibilidad de información relacionada con los estudiantes. Esta aplicación permite a los alumnos agendar citas con los capellanes, quienes pueden organizar y acceder fácilmente a los datos recopilados durante las entrevistas y encuestas, todo ello sin el uso excesivo de papel y con un enfoque más ecológico. La arquitectura MVC se descompone en tres componentes: el modelo, que maneja la lógica de negocio; la vista, que presenta la interfaz gráfica al usuario; y el controlador, que actúa como intermediario entre los otros dos. Este sistema está diseñado para adaptarse fácilmente, permitiendo la implementación futura de nuevas funciones y características, todo mientras se mantiene el código modular y fácil de mantener. A través del uso de AJAX, Bootstrap y PHP, la aplicación promete flexibilidad y facilidad de uso para los capellanes, mejorando la eficiencia y el alcance de sus actividades de consejería.

Reflexión:

Este artículo deja como reflexión la importancia de adaptar las herramientas tecnológicas a las necesidades específicas de instituciones educativas y religiosas, como el sistema de capellanía. La aplicación del patrón MVC no solo proporciona una estructura clara y fácil de mantener, sino que también permite a los capellanes enfocarse en la orientación espiritual y académica de los estudiantes sin verse limitados por herramientas arcaicas. Este modelo también muestra cómo la tecnología puede humanizar y hacer más accesible la orientación personal al permitir un seguimiento detallado y eficiente de cada estudiante. Además, enfatiza que la tecnología, cuando es bien aplicada, puede facilitar el trabajo de los capellanes y promover una mejor relación con los estudiantes. También destaca la importancia de implementar medidas de seguridad, como la encriptación de contraseñas y autenticación, para proteger la información de los usuarios. Este enfoque, además de ser ecológico al reducir el uso de papel, muestra cómo un sistema modular y bien diseñado puede servir de base para futuras actualizaciones, fomentando un entorno educativo y pastoral más eficiente, accesible y sostenible.

Diagrama:



Bbliografía:

Yair P. Implementación del patrón arquitectónico MVC en aplicaciones web para la arquitectura del software del sistema de capellanía de la UM. *Anuario de Investigación UM*. 2020;1(1):112-120. Accessed November 19, 2024.

https://n9.cl/eii0f2

ARTICULO 23

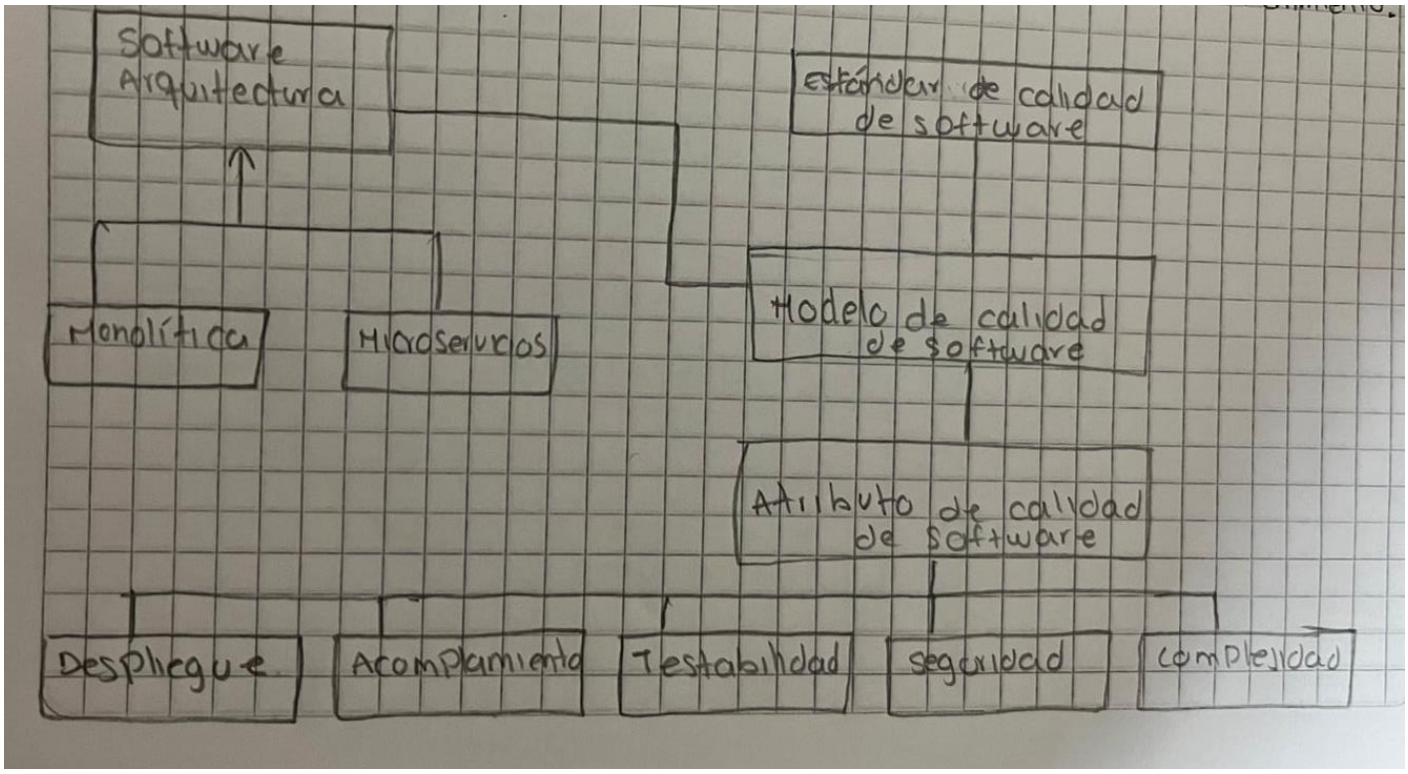
Desarrollo de un modelo basado en la calidad para la optimización de la arquitectura de software: un estudio de caso de arquitecturas monolíticas y de microservicios

El artículo presenta un modelo centrado en la calidad para optimizar la arquitectura de software, comparando arquitecturas monolíticas y de microservicios. Evalúa atributos de calidad como el acoplamiento, la testabilidad, la seguridad, la complejidad, la capacidad de despliegue y la disponibilidad, usando un caso de estudio con Jakarta EE y Spring. El estudio propone un modelo matemático para tomar decisiones arquitectónicas que optimicen la calidad, permitiendo combinar elementos monolíticos y de microservicios según las necesidades del sistema. Este enfoque destaca la importancia de reducir dependencias en microservicios y mantener la complejidad manejable en entornos monolíticos. El artículo concluye con recomendaciones prácticas, señalando que las decisiones arquitectónicas deben priorizar los atributos de calidad y someterse a evaluaciones continuas para cumplir con los requisitos no funcionales. Esto permite que las decisiones de arquitectura se alineen con los objetivos y necesidades a largo plazo del sistema, favoreciendo la sostenibilidad y adaptabilidad del software.

Reflexión:

Este artículo invita a reflexionar sobre la arquitectura de software no solo como una elección técnica, sino como una decisión estratégica que impacta profundamente en la calidad, sostenibilidad y adaptabilidad del sistema a largo plazo. Resalta la importancia de contar con una arquitectura flexible que pueda ajustarse a los cambios y necesidades emergentes del proyecto, equilibrando simplicidad en diseños monolíticos y escalabilidad en microservicios mediante optimizaciones basadas en la calidad. En este sentido, el enfoque de calidad propuesto permite a los desarrolladores tomar decisiones informadas que no solo optimicen el rendimiento y la seguridad, sino que también respondan a las expectativas de los usuarios.

Diagrama:



Bibliografía:

Miloš Milić, Dragana Makajić-Nikolić. Development of a Quality-Based Model for Software Architecture Optimization: A Case Study of Monolith and Microservice Architectures. *Symmetry*. 2022;14(9):1824-1824. doi:https://doi.org/10.3390/sym14091824

ARTICULO 24

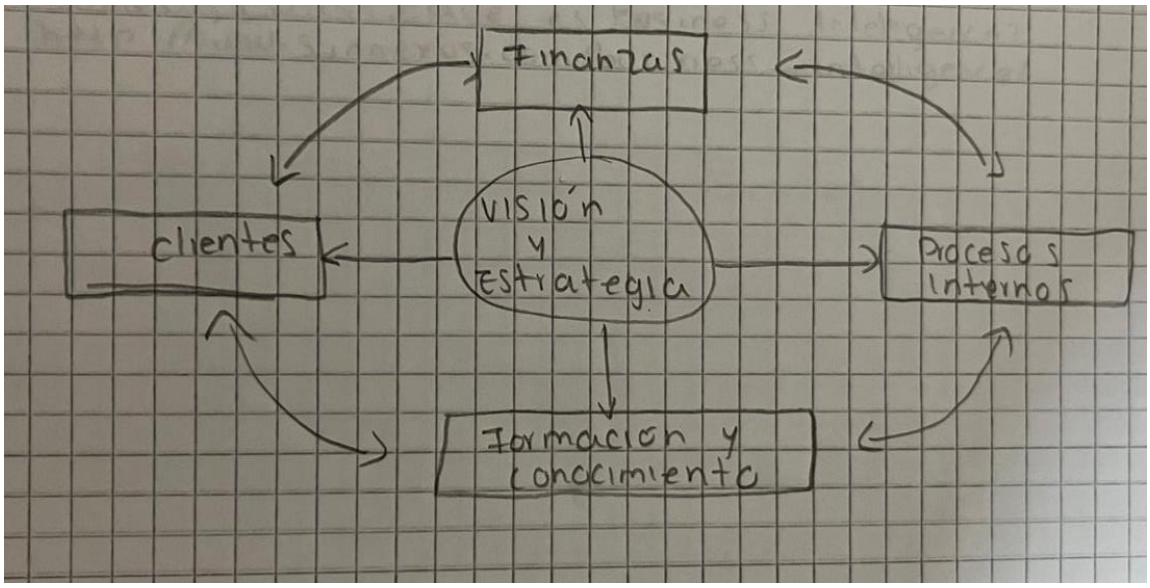
Arquitectura de software para la construcción de un sistema de cuadro de mando integral como herramienta de inteligencia de negocios.

Este artículo explora el papel crucial de la arquitectura de software en el diseño de sistemas de inteligencia de negocios, específicamente a través del cuadro de mando integral (CMI) como una herramienta para optimizar la toma de decisiones empresariales. El CMI permite transformar los datos generados en una organización en información relevante y conocimiento útil, lo cual es vital para apoyar las decisiones estratégicas. Utilizando un datawarehouse, la arquitectura permite recopilar y estructurar datos de varias fuentes, mejorando la accesibilidad y eficiencia en el análisis. Además, se destaca cómo el uso de modelos como el de Kimball facilita la creación de una infraestructura de datos robusta y flexible, permitiendo integrar, almacenar y analizar información de manera eficiente. En el contexto del CMI, se consideran múltiples perspectivas —financiera, de clientes, de procesos internos y de aprendizaje—, con el fin de equilibrar los objetivos a corto y largo plazo. Este enfoque estructurado contribuye a que las empresas puedan evaluar sus estrategias de manera constante y adaptar sus procesos a los cambios en el entorno empresarial, aprovechando la tecnología para optimizar su rendimiento y crecimiento sostenido.

Reflexión:

Este artículo deja como reflexión que la arquitectura de software y las herramientas de inteligencia de negocios no solo se limitan a mejorar el análisis de datos, sino que también transforman el enfoque estratégico de las organizaciones hacia una visión integrada y adaptable. La implementación de un sistema de cuadro de mando integral permite que las empresas gestionen sus recursos de forma óptima, evaluando continuamente el impacto de sus decisiones en varias áreas clave, como finanzas, procesos internos, clientes y aprendizaje. Además, estos sistemas no solo funcionan como herramientas para organizar datos, sino que facilitan la generación de información valiosa que puede convertirse en conocimiento estratégico.

Diagrama:



Bibliografía:

Adolfo G. Arquitectura de software para la construcción de un sistema de cuadro de mando integral como herramienta de inteligencia de negocios. *Tecnología Investigación y Academia*. 2017;5(2):143-152. Accessed November 19, 2024.

https://revistas.udistrital.edu.co/index.php/tia/article/view/8766

ARTICULO 25

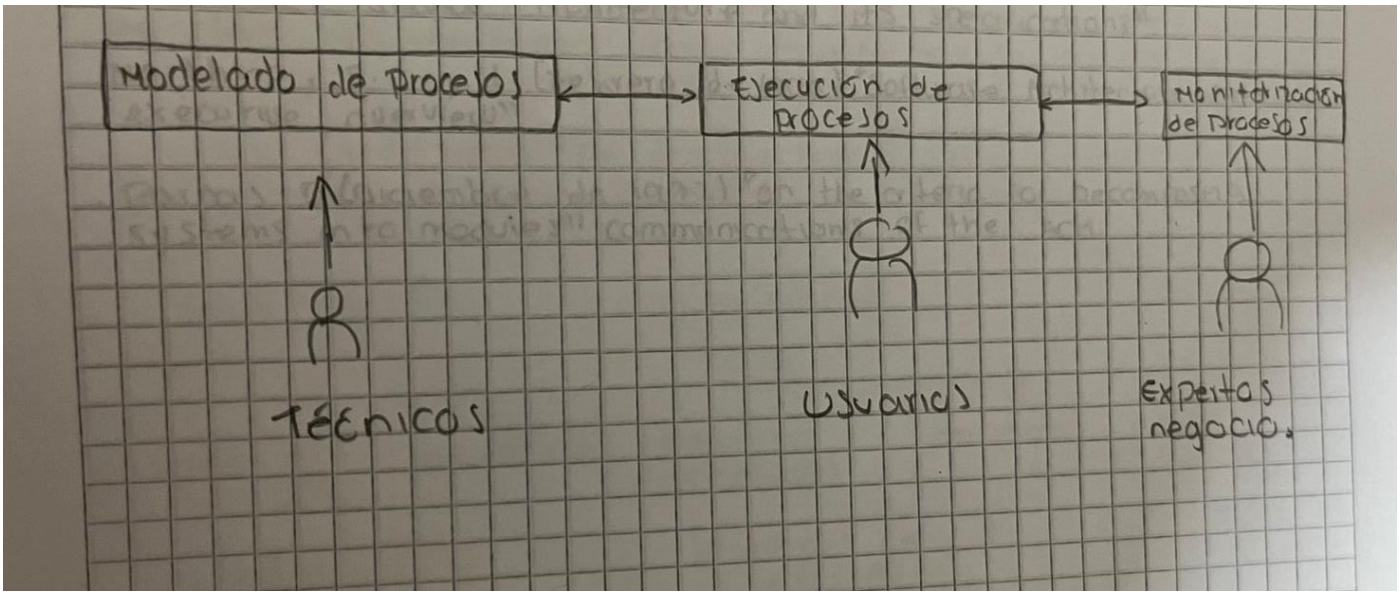
Arquitectura de software. Arquitectura orientada a servicios

El artículo aborda la Arquitectura Orientada a Servicios (SOA) como una filosofía de diseño clave en la evolución de la arquitectura de software. En su contexto histórico, se mencionan hitos como los principios de modularidad y ocultamiento de información de Parnas y la analogía de la arquitectura con la construcción de edificios de Perry y Wolf. SOA se destaca por permitir la interoperabilidad entre sistemas heterogéneos, la separación lógica de procesos y la alineación con las necesidades del negocio. La implementación de SOA se basa en estándares abiertos que fomentan la reutilización de servicios, la integración eficiente de aplicaciones y la flexibilidad frente a cambios tecnológicos. Sus beneficios abarcan la mejora en la toma de decisiones mediante el acceso a información actualizada, el aumento de la productividad de empleados y la optimización de las relaciones con clientes y proveedores. Además, se describen conceptos como los BPM (Business Process Management), que refuerzan la automatización y el modelado de procesos, y elementos esenciales como el Bus de Servicios y los repositorios, que soportan la ejecución y monitorización de servicios. SOA trasciende los límites tradicionales del desarrollo de software, consolidándose como un paradigma esencial para el éxito organizacional en la era digital.

Reflexión:

Este artículo deja como reflexión que la Arquitectura Orientada a Servicios (SOA) representa un cambio significativo en cómo las organizaciones abordan el desarrollo de software, enfocándose en la agilidad, la integración y la escalabilidad. SOA no solo ofrece una solución técnica para optimizar procesos, sino que también redefine cómo se alinean los objetivos tecnológicos con las estrategias de negocio. La capacidad de reutilizar componentes y servicios proporciona a las empresas una ventaja competitiva, permitiéndoles adaptarse rápidamente a las demandas del mercado. Además, el enfoque en estándares abiertos y la interoperabilidad promueve un entorno colaborativo donde los sistemas pueden evolucionar de manera orgánica.

Diagrama:



Bibliografía:

Yanet Espinal Martín. Arquitectura de software. Arquitectura orientada a servicios. ***Serie Científica de la Universidad de las Ciencias Informáticas*****. 2024;5(1):1-10.**

doi:https://dialnet.unirioja.es/descarga/articulo/8590088.pdf

ARTICULO 26

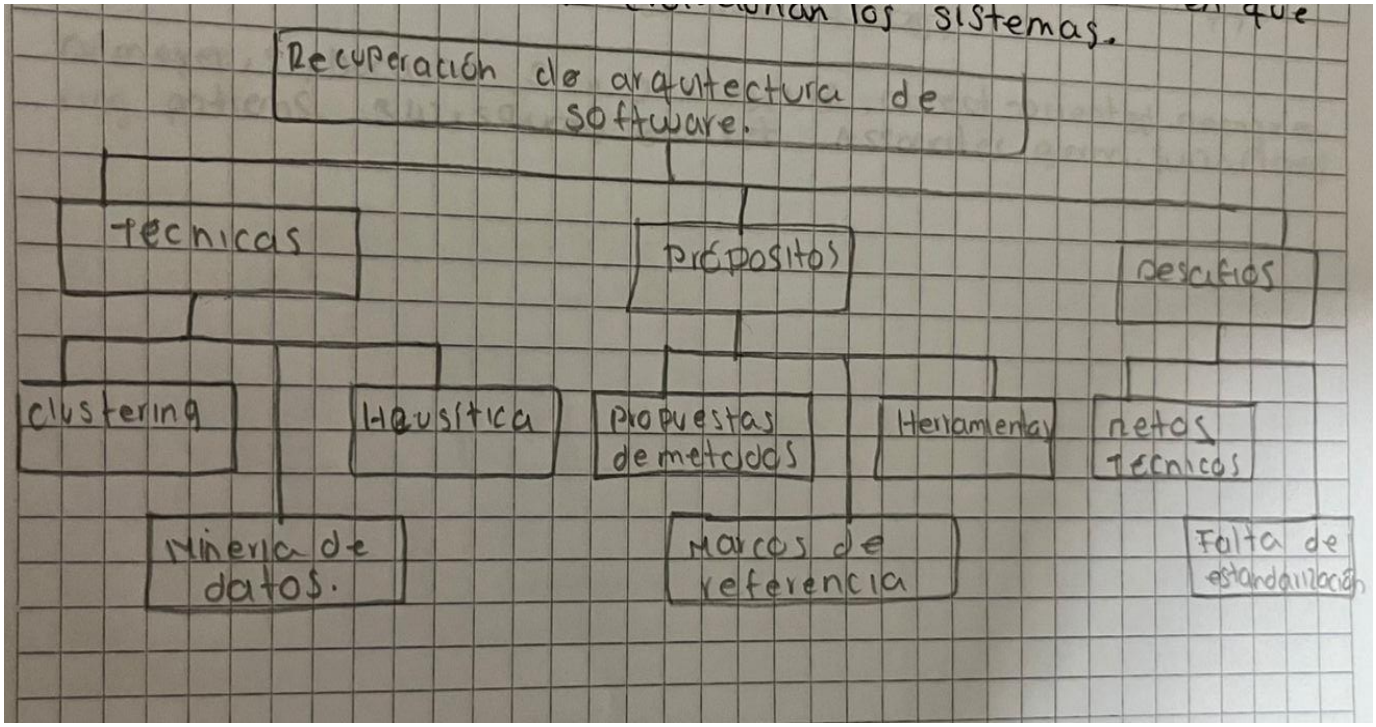
Recuperación de Arquitecturas de Software: Un Mapeo Sistemático de la Literatura

El artículo aborda un mapeo sistemático de la literatura sobre la recuperación de arquitecturas de software, un proceso crucial para mejorar el entendimiento y mantenimiento de sistemas complejos. Mediante la recopilación y análisis de 4050 documentos, seleccionados bajo estrictos criterios de inclusión y exclusión, se identificaron tendencias en técnicas como clustering, heurística y minería de datos. El propósito principal de los estudios evaluados es conceptual, enfocado en proponer métodos, herramientas y marcos de referencia. La investigación resalta desafíos como la falta de estandarización en la representación de vistas arquitectónicas recuperadas y los retos técnicos para integrar resultados de distintas propuestas. Se destacan modelos de representación basados en grafos, UML y lenguajes de descripción como ACME. Finalmente, se enfatiza la necesidad de un enfoque unificado para facilitar la reutilización y análisis de arquitecturas, proponiendo un catálogo de técnicas que detalle cuándo y cómo emplearlas para maximizar su utilidad en contextos específicos.

Reflexión:

Este artículo deja como reflexión la importancia de abordar los desafíos en la recuperación de arquitecturas de software desde una perspectiva integrada que fomente la colaboración entre investigadores y la industria. La falta de estandarización en las técnicas y representaciones plantea barreras significativas para el avance en el campo, limitando la reutilización de conocimiento y la interoperabilidad entre herramientas. La propuesta de un catálogo unificado es un paso necesario hacia la consolidación de un cuerpo teórico sólido, pero su éxito dependerá de su adopción práctica y validación en escenarios reales. Asimismo, la creciente complejidad de los sistemas modernos exige enfoques más robustos y adaptativos que combinen técnicas dinámicas y estáticas para capturar tanto la estructura como el comportamiento del software.

Diagrama:



Bibliografía:

Monroy ME, Arciniegas JL, Rodríguez JC. Recuperación de Arquitecturas de Software: Un Mapeo Sistemático de la Literatura. *Información tecnológica*. 2016;27(5):201-220. doi:https://doi.org/10.4067/s0718-07642016000500022

ARTICULO 27

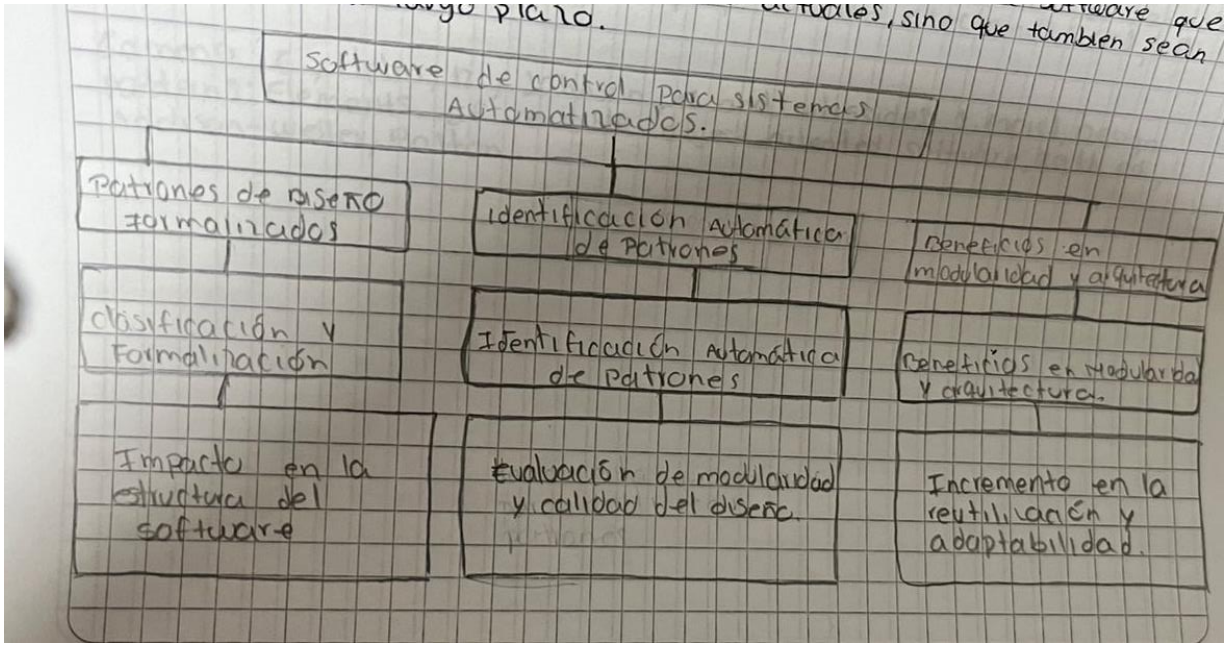
Formalización de patrones de diseño y su identificación automática en software PLC para evaluación de arquitectura.

El artículo trata sobre la formalización e identificación automática de patrones de diseño en el software de control para sistemas de producción automatizados (aPS). Con la creciente complejidad de estos sistemas, provocada por la personalización masiva y la diversidad de productos, surge la necesidad de estrategias que permitan una modularización efectiva y la reutilización del software. Se presenta un enfoque que clasifica y formaliza patrones estructurales comunes en el software de control, evaluando su impacto en el modularidad y la arquitectura del sistema. La implementación de un prototipo demuestra cómo estos patrones pueden ser identificados automáticamente, facilitando la creación de software más modular, reutilizable y adaptable. Asimismo, se discuten los beneficios de aplicar estos patrones en entornos industriales, como la mejora en la calidad del diseño y la reducción de problemas estructurales. Su uso permite identificar áreas de mejora en la arquitectura de software, optimizando el proceso de desarrollo y promoviendo la sostenibilidad del sistema frente a cambios y escalabilidad futura.

Reflexión:

Este artículo deja como reflexión que la formalización de patrones de diseño en el software de sistemas industriales no solo contribuye a la modularización, sino que también fomenta una arquitectura más sostenible y escalable. La capacidad de identificar automáticamente patrones proporciona un enfoque más sistemático, minimizando errores y optimizando la calidad del diseño. La integración de estas prácticas en entornos industriales representa un avance hacia una ingeniería de software más eficiente, adaptable y reutilizable. Además, sugiere que el modularidad no es únicamente una ventaja técnica, sino también estratégica, ya que permite a las organizaciones responder con mayor agilidad a los cambios del mercado y a las crecientes demandas de personalización. Adopciones como estas son esenciales para asegurar un desarrollo tecnológico sostenible en el ámbito de la producción automatizada.

Diagrama:



Bibliografía:

Neumann EM, Vogel-Heuser B, Fischer J, Ocker F, Diehm S, Schwarz M. Formalization of Design Patterns and Their Automatic Identification in PLC Software for Architecture Assessment. *IFAC-PapersOnLine*. 2020;53(2):7819-7826. doi:https://doi.org/10.1016/j.ifacol.2020.12.1881

ARTICULO 28

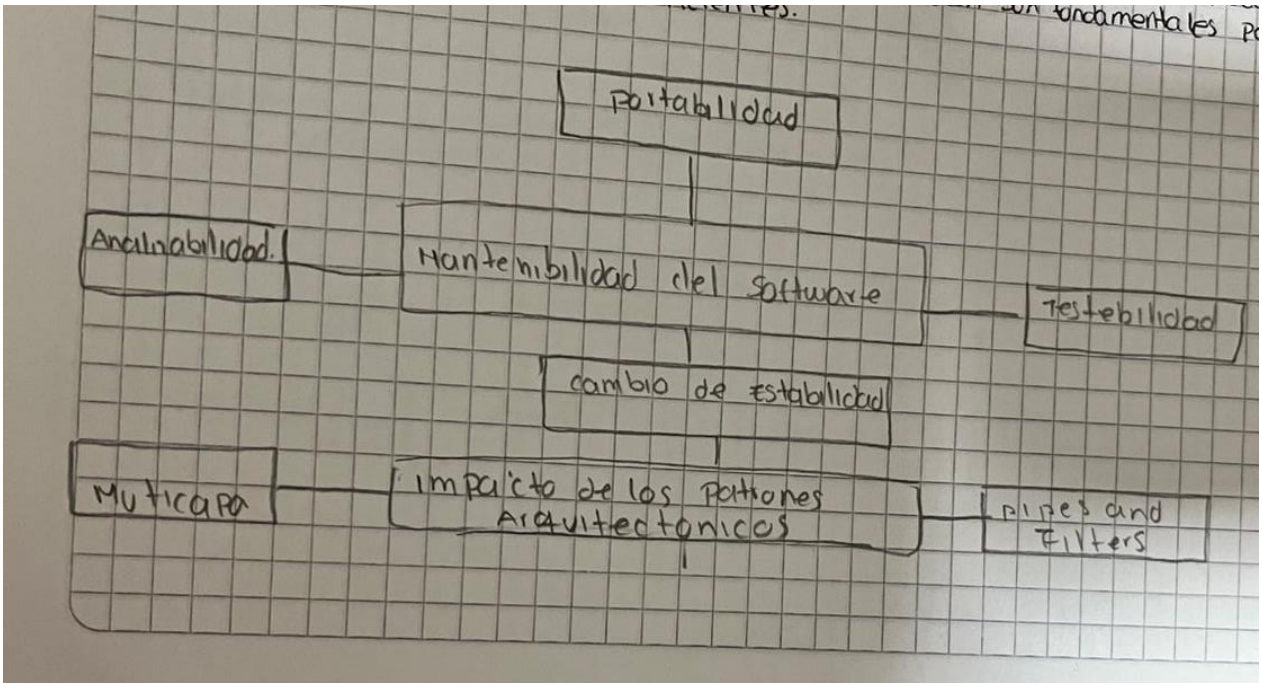
Garantizar la mantenibilidad del software a nivel de arquitectura de software utilizando patrones arquitectónicos.

El artículo analiza la importancia de la arquitectura de software en la mejora de la mantenibilidad del software, destacando cómo los patrones arquitectónicos influyen directamente en atributos clave como analizabilidad, capacidad de cambio, estabilidad, testabilidad, comprensibilidad y portabilidad. Se propone un modelo de calidad para evaluar la mantenibilidad a nivel arquitectónico, dividiendo los atributos en subatributos medibles mediante métricas específicas. Además, se exploran patrones arquitectónicos comunes, como el patrón multicapa y el patrón pipes and filters, evaluando su impacto en la mantenibilidad. El artículo también discute experiencias reales de refactorización arquitectónica, demostrando cómo una arquitectura adecuada puede reducir costos de mantenimiento y mejorar la eficiencia en el desarrollo de software. Se enfatiza que, aunque un patrón bien elegido facilita la mantenibilidad, una implementación inadecuada puede tener efectos negativos. Finalmente, se destacan los desafíos actuales en la evaluación de costos de refactorización y la necesidad de herramientas más precisas para alinear los patrones arquitectónicos con los requisitos del cliente.

Reflexión:

Este artículo deja como reflexión que la arquitectura de software desempeña un papel crítico en la sostenibilidad y evolución de sistemas informáticos. La elección adecuada de patrones arquitectónicos no solo facilita la mantenibilidad, sino que también mejora aspectos clave como la eficiencia y la adaptabilidad. Sin embargo, este proceso requiere un análisis meticuloso para evitar efectos adversos que puedan surgir de decisiones arquitectónicas inadecuadas. Además, la investigación subraya la necesidad de un enfoque proactivo en la planificación arquitectónica, integrando herramientas que permitan medir y optimizar atributos de calidad desde las etapas iniciales del diseño. La experiencia de refactorización presentada destaca cómo la transición hacia una arquitectura más estructurada puede transformar la experiencia de mantenimiento, ahorrando tiempo y recursos.

Diagrama:



Bibliografía:

Rahmati Z, Tanhaei M. Ensuring software maintainability at software architecture level using architectural patterns. 2021;2(1):81-102. doi:https://doi.org/10.22060/ajmc.2021.19232.1044

ARTICULO 29

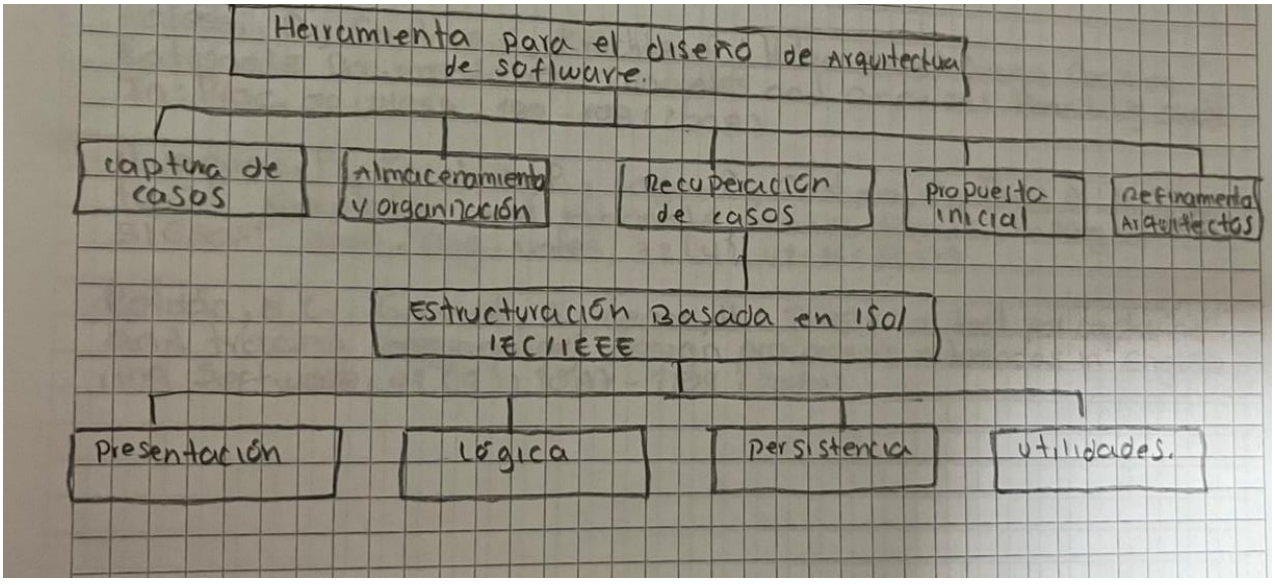
Una herramienta para reutilizar estrategias en diseños de arquitecturas de software.

El artículo presenta RADS, una herramienta innovadora para el diseño de arquitecturas de software que se enfoca en la reutilización de experiencias pasadas de arquitectos. Basándose en el razonamiento basado en casos, RADS permite capturar, almacenar, recuperar y reutilizar estrategias de diseño arquitectónico previas para resolver nuevos problemas. La herramienta organiza la información en casos arquitectónicos estructurados según el estándar ISO/IEC/IEEE 42010:2011 y utiliza escenarios específicos del sistema y restricciones de diseño para identificar similitudes entre casos pasados y actuales. RADS opera en cuatro capas: presentación, lógica, persistencia y utilidades, garantizando flexibilidad e integración con herramientas UML existentes como Eclipse. Además, ofrece funcionalidades como la gestión de atributos de calidad, estrategias de diseño y escenarios arquitectónicos. Los casos recuperados se utilizan para generar propuestas iniciales que luego son refinadas por los arquitectos. Este enfoque promueve la sostenibilidad en el diseño arquitectónico, ahorra tiempo y facilita la colaboración entre arquitectos, posicionando a RADS como una herramienta estratégica en la ingeniería de software

Reflexión:

Este artículo deja como reflexión que la reutilización en el diseño de arquitecturas de software no solo optimiza el tiempo y los recursos, sino que también fomenta la sostenibilidad y la colaboración en la industria tecnológica. RADS demuestra que aprovechar las experiencias pasadas y estructurarlas en un marco lógico puede transformar la manera en que se abordan los problemas de diseño. La integración con estándares internacionales y herramientas existentes resalta la importancia de alinear las innovaciones con prácticas establecidas para maximizar su adopción y efectividad. Además, RADS refuerza la idea de que el conocimiento compartido es un activo invaluable para los arquitectos, proporcionando una base sólida para la toma de decisiones en etapas tempranas del desarrollo.

Diagrama:



Bibliografía:

Celeste CM, Gonnet, Silvio M, Leone HP. RADS: una herramienta para reutilizar estrategias en diseños de arquitecturas de software. *Unlpeduar*. Published online November 30, 2016. doi:http://sedici.unlp.edu.ar/handle/10915/57164

ARTICULO 30

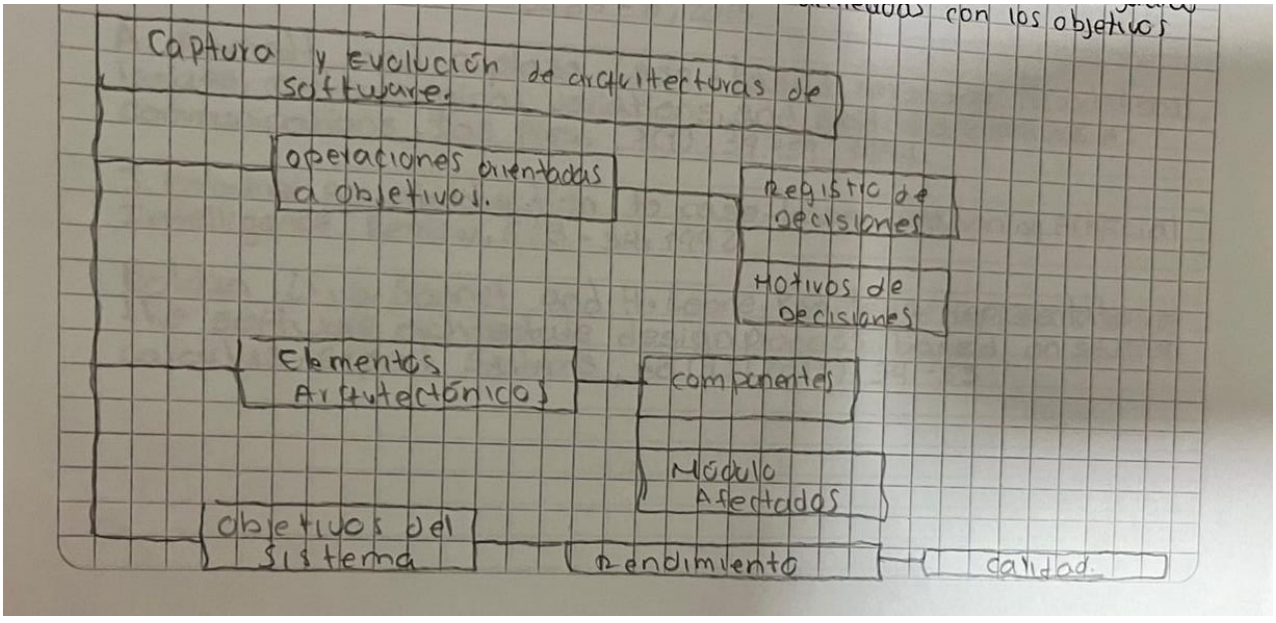
Captura del razonamiento y evolución de arquitecturas de software mediante la aplicación de operaciones arquitectónicas orientadas a objetos..

El artículo aborda un modelo para capturar y documentar la evolución y el razonamiento en el diseño de arquitecturas de software, un desafío crucial en el desarrollo de sistemas complejos. La propuesta se centra en operaciones orientadas a objetivos (OpOOs), que registran no solo las decisiones tomadas, sino también los motivos detrás de ellas. Este modelo organiza las decisiones como operaciones aplicables a elementos arquitectónicos, vinculándolas con objetivos como calidad, rendimiento o restricciones específicas. Esto mejora la trazabilidad entre las decisiones, los productos arquitectónicos generados y los objetivos alcanzados. También facilita la comprensión de cómo y por qué una arquitectura evolucionó de cierta manera. Además, se proponen herramientas para la gestión y sugerencia de operaciones adecuadas al contexto, lo que ayuda a los arquitectos a optimizar el proceso de diseño y documentación. Se ilustra su uso mediante la migración de un sistema a la nube híbrida, mostrando cómo se satisfacen requisitos de escalabilidad y rendimiento mediante operaciones planificadas. El modelo representa una contribución significativa al diseño arquitectónico al sistematizar procesos y reducir la pérdida de conocimiento durante la evolución de sistemas.

Reflexión:

Este artículo deja como reflexión que la documentación y trazabilidad en el diseño de arquitecturas de software no solo son necesarias, sino también posibles mediante enfoques sistemáticos y herramientas adecuadas. La incorporación de operaciones orientadas a objetivos representa un avance al permitir que los arquitectos documenten sus decisiones y razonamientos de manera más natural y eficiente, sin aumentar significativamente su carga de trabajo. Esto no solo mejora la calidad del diseño actual, sino que también facilita futuras modificaciones y evaluaciones, fomentando prácticas más responsables y sostenibles. Además, el enfoque destaca la importancia de considerar metas y restricciones específicas desde el inicio del proceso, asegurando que las decisiones arquitectónicas estén alineadas con los objetivos del sistema.

Diagrama:



Bibliografía:

Luciana RM, Gonnet, Silvio M, Leone HP. Captura del razonamiento y evolución de arquitecturas de software mediante la aplicación de operaciones arquitectónicas orientadas a objetivos. *Unlpeduar*. Published online 2015. doi:http://sedici.unlp.edu.ar/handle/10915/52407