Soy Mariana Charry Prada, una persona comprometida con mi crecimiento profesional y personal. Actualmente, me encuentro cursando el programa de "*Análisis y Desarrollo de Software"* en el "*Servicio Nacional de Aprendizaje" (SENA),* en Neiva, Huila; un paso más en mi formación técnica que comenzó con el título de *Bachiller Técnico en Sistemas* en el "*Colegio Adventista Baluarte Interamericano".* En 2022, culminé mi *Técnico en "Programación de Software"* en el SENA, lo que me permitió iniciar mi conocimiento del desarrollo de software y mis habilidades técnicas.

Me considero una persona responsable, dinámica y con un fuerte deseo de superación. Siempre busco aprender de manera rápida y eficiente, y tengo un gran compromiso con la puntualidad, la honestidad y la responsabilidad en todas las actividades que realizo. A lo largo de mi formación y mis experiencias, he desarrollado una capacidad para enfrentar desafíos con una actitud positiva y proactiva.

Tengo una gran capacidad de trabajo en equipo, pero también soy autónoma y organizada cuando se trata de realizar tareas individuales. Me gusta asumir la responsabilidad de mis proyectos y cumplir con los objetivos establecidos, siempre manteniendo un alto estándar de calidad en mi trabajo.

Mi capacidad para adaptarme a diferentes entornos y mi sentido de pertenencia son dos de mis características más destacadas. Valoro profundamente el trabajo colaborativo y la posibilidad de aportar al éxito colectivo, por lo que me esfuerzo por ser una persona en la que los demás confíen y con la que disfruten trabajar.

Artículo 1:

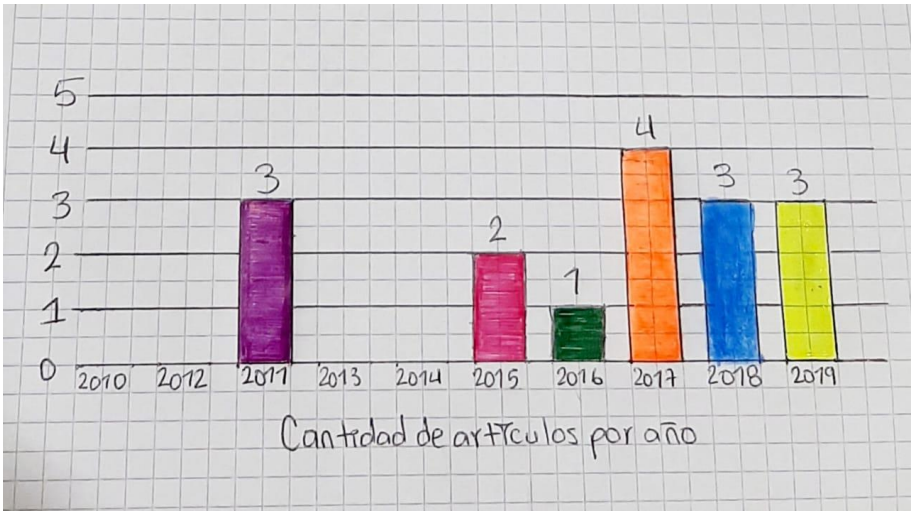
Título:

Análisis comparativo de patrones de diseño de software para el desarrollo de aplicaciones móviles de calidad: Una revisión sistemática de la literatura

Resumen:

En el inmenso mundo de las aplicaciones móviles existen diversos patrones de diseño que nos brindan facilidades de crear un desarrollo más eficiente y organizado. El objetivo de la revisión es el de encontrar los principales estudios sobre patrones de diseño de software para el desarrollo de aplicaciones móviles de calidad, y posteriormente determinar criterios de identificación que servirán como herramienta de selección de patrones de diseño de calidad.

Gráfica:



Reflexión:

Los patrones de diseño han mostrado u mayor impacto en el desarrollo de software. Dado que los patrones de diseño proporcionan soluciones comprobadas, el desarrollo de aplicaciones móviles puede ser un proceso tedioso, ya que cada aplicación debe pasar por los ciclos de desarrollo para garantizar que la aplicación se ajuste a los atributos de calidad estándar. Luego de que el desarrollador compare los principales patrones de diseño de software, finalmente podrá implementar la más conveniente para el proyecto.

Cita APA:

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.

Artículo 2:

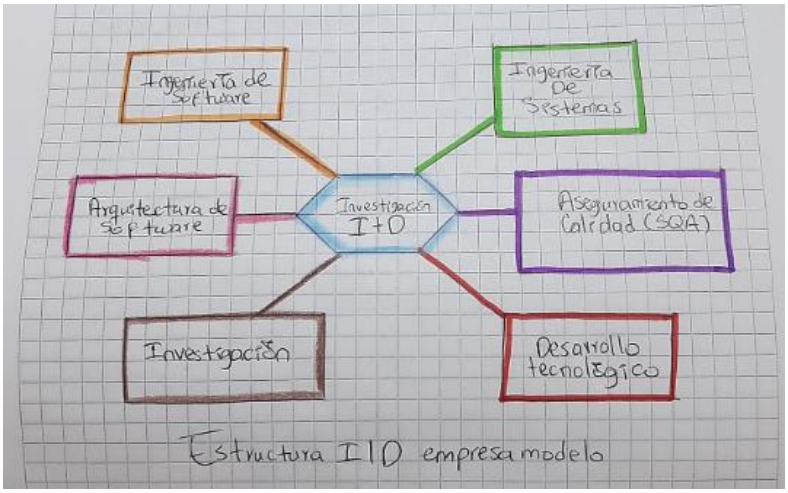
Título:

Recomendaciones para la Formación de una Empresa de Desarrollo de Software Competitiva en un País como Colombia

Resumen:

Algunas de las recomendaciones para ser una empresa competitiva, son, buscar que la organización sea reconocida como una compañía transparente con los clientes, la estructura de la organización, debe estar fortalecida con un organismo que oriente el desarrollo tecnológico tener mecanismos que les permitan medir y saber si se están cumpliendo las metas establecidas en los procesos, los productos y proyectos de la compañía.

Gráfica:



Reflexión:

Al tener una empresa dirigida al desarrollo de software, o al querer formarla, es necesario tener ciertos criterios importantes para de esa manera poder ser una empresa competitiva en todo este campo del desarrollo de software, más en un país como lo es Colombia, el cual en estos tiempos a tomado fuerza en el tema de la tecnología y sus componentes. Todo el artículo hablaba sobre recomendaciones, las nombraban y luego explicaban cada una de ellas, para que la organización de una compañía sea buena y resalte.

Cita APA:

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.

Artículo 3:

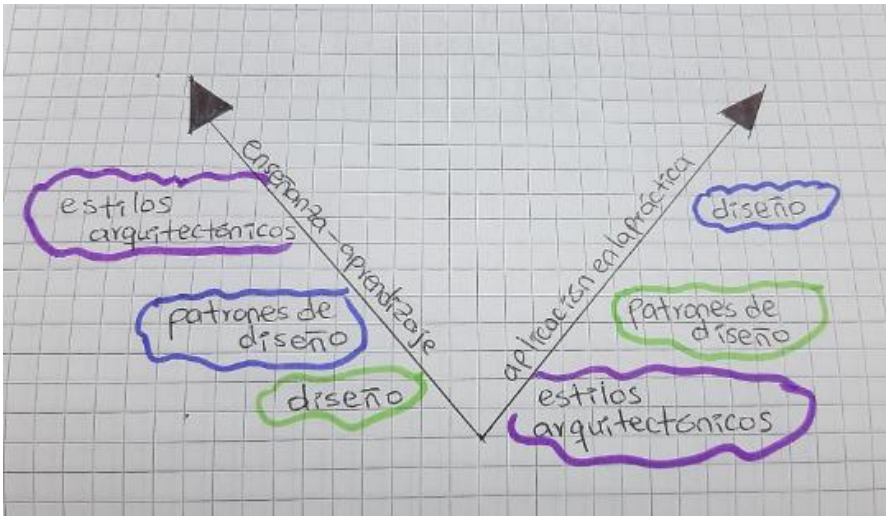
Título:

Introducción a la Arquitectura de Software

Resumen:

En los primeros años de la construcción de software no existía el diseño del sistema como una etapa independiente a la programación. A principios de la década del 90 algunos investigadores, comenzaron a ver la necesidad de investigar y desarrollar un nivel de abstracción superior al dl diseño, al que llamaron "Arquitectura de software". La arquitectura del sistema es el resultado de combinar decisiones técnicas, sociales y del negocio.

Gráfica:



Reflexión:

A partir de la creación de la arquitectura de software, se crearon de igual manera diferentes y variados modelos de desarrollo, como el "Modelo de Cascada", la arquitectura del sistema, la cual es la que se le sigue a la Ingeniería de Requerimientos. La fase de la arquitectura del sistema, se subdivide en tres etapas: elección del estilo arquitectónico, selección de los patrones de diseño y diseño de componentes; lo cual es importantes de aprender para llevarla a la práctica, también es importante conocer los estilos arquitectónicos, estos son una generalización de los patrones de diseño, como abstracción.

Cita APA:

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.

Artículo 4:

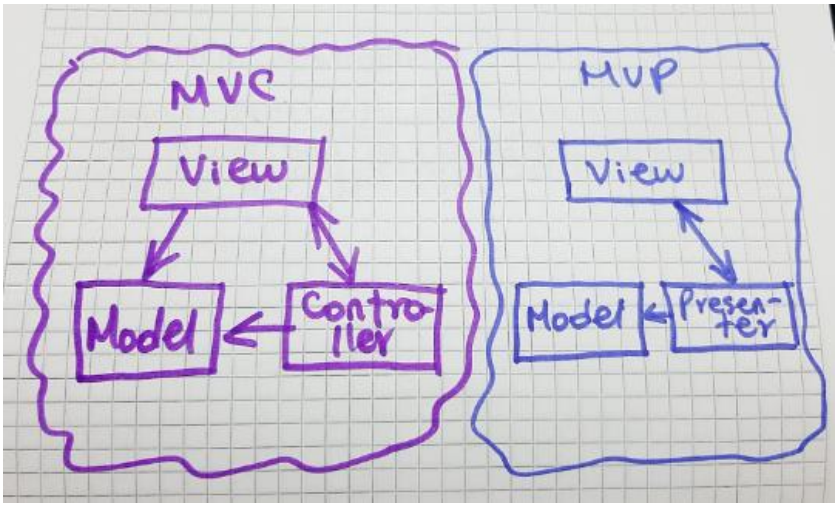
Título:

Lenguajes de Patrones de Arquitectura de Software: Una Aproximación al Estado del Arte

Resumen:

Mostrar el estado del arte en un área de la arquitectura de software llamada "Lenguajes de Patrones", desde sus orígenes los avances actuales y sus aplicaciones en la construcción de arquitecturas de software en diferentes dominios de aplicación. La extensibilidad y aplicabilidad de los lenguajes de patrones, se convierten en una herramienta importante para diseñadores e implementadores de todo tipo de sistemas de información. Es importante evitar la programación al estilo vaquero. Parches de último minuto o trabajo de última noche que pueden traer graves consecuencias.

Gráfica:



Reflexión:

Para que un software sea bueno y tenga una funcionalidad correcta, es importante evitar hacer las cosas demasiado rápido, porque se pueden presentar especificaciones incompletas, lo cual puede resultar en un re-proceso, haciendo perder tiempo en la elaboración.

La creación de nuevas formas de hacer las cosas ayuda en la productividad y una mayor calidad en los productos de software.

Cita APA:

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.

Artículo 5:

Título:

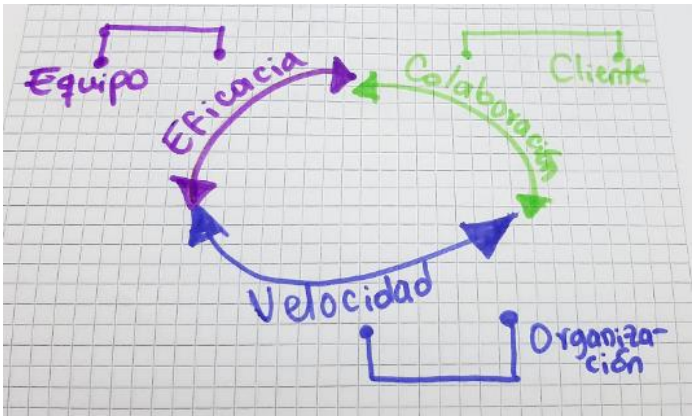
Integración de arquitectura de software en el ciclo de vida de las metodologías ágiles

Resumen:

Las metodologías ágiles se centran en el trabajo en equipo, la adaptabilidad y colaboración dentro del grupo de software y también entre los miembros del grupo y los usuarios finales. El uso de las metodologías (MA), ha marcado una tendencia en su adopción al desarrollo de proyectos de software (AR), en tanto, es una manifestación de decisiones de etapas muy tempranas del diseño sobre un sistema.

El tratamiento con enfoques diferentes en las primeras etapas, ha sido uno de los factores que ha causado la sensación de las MA y la AS.

Gráfica:



Reflexión:

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

Cita APA:

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).

Artículo 6:

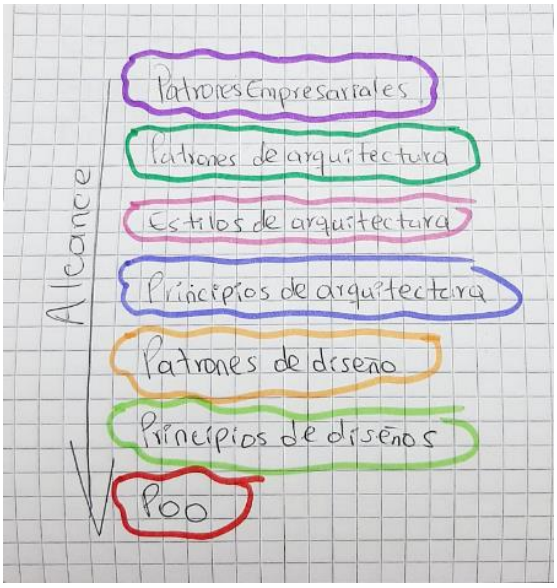
Título:

Evaluación de una Arquitectura de Software

Resumen:

Una arquitectura de software es clave para que las organizaciones puedan avanzar y concentrarse en su función misional, solo si esta está bien diseñada. Es importante realizar evaluaciones tempranas a la arquitectura, mediante algún método, porque eso ayuda a la elección de una arquitectura, mejora la comunicación y permite una mejor interpretación de las historias de usuario. También tener en cuenta el análisis para el atributo de calidad modificabilidad.

Gráfica:



Reflexión:

La arquitectura de software es el nacimiento sobre el que 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. En este contexto, realizar evaluaciones tempranas y continuas de la arquitectura es crucial. Cuando se elige una arquitectura adecuada desde el principio, se pueden evitar costosos rediseños a medida que el sistema crece o se modifica.

Cita APA:

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

Artículo 7:

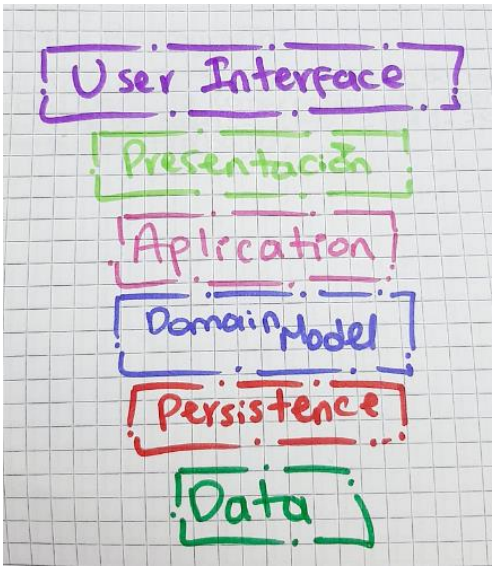
Título:

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

Resumen:

El desarrollo de software implica considerar una cantidad variada de aspectos tecnológicos. Entre los más destacados podemos mencionar los relacionados con el acceso a datos, las interfaces, los procesos funcionales, el control de las transacciones, la accesibilidad y la seguridad. Lograr un diseño coherente con los requerimientos planteados, niveles aceptables de flexibilidad, extensibilidad y usabilidad, así como facilitar las actividades de mantenimiento lleva a pensar la concepción del software en capas.

Gráfica:



Reflexión:

El diseño en capas 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 forma independiente. Por ejemplo, si se necesita cambiar el tipo de base de datos o actualizar un framework, podemos hacerlo sin alterar la lógica de negocio ni la interfaz de usuario.

Cita APA:

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

Artículo 8:

Título:

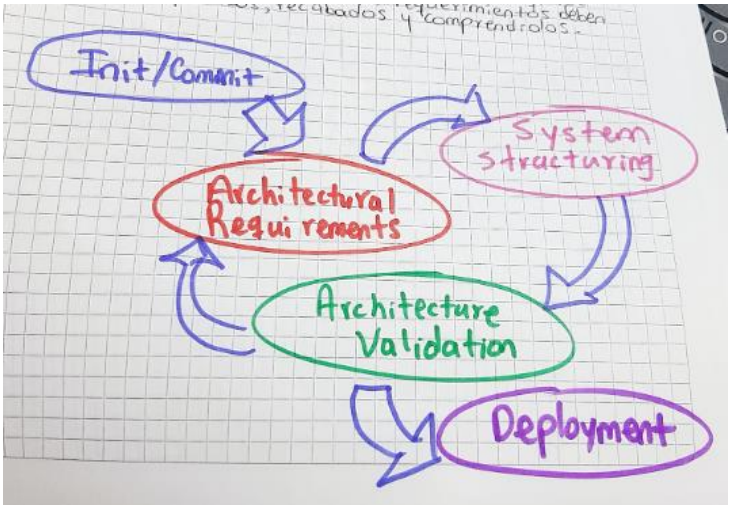
Arquitectura de software en el proceso de desarrollo ágil: una perspectiva basada en requisitos significantes para la arquitectura.

Resumen:

Recopilar, comprender y gestionar los requisitos es un aspecto crítico en todos los métodos de desarrollo. Esto también es cierto para las metodologías ágiles en la que la captura de requisitos es realizada en todo el proceso de desarrollo, con requisitos que van evolucionando y cambiando a lo largo del ciclo de vida.

Este proceso es opuesto al enfoque de la arquitectura del software, donde los requerimientos deben ser identificados, recabados y comprendidos.

Gráfica:



Reflexión:

La reflexión sobre la gestión de requisitos en las metodologías ágiles frente al enfoque tradicional de la arquitectura de software revela una de las tensiones más fundamentales en el desarrollo de software. En un enfoque tradicional o "en cascada", la captura y definición de requisitos es una fase crítica inicial. Una vez que estos requisitos son comprendidos y documentados, la arquitectura del software se diseña y se ajusta para cumplir con esos requisitos. En este escenario, los requisitos se entienden como algo fijo, que debe ser meticuloso.

Cita APA:

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).

Artículo 9:

Título:

La Arquitectura de Información (AI) en el proceso de desarrollador de Software.

Resumen:

La AI se ha convertido para la producción de software en un proceso determinante con vistas a que los productos alcancen la calidad requerida. El objetivo de la investigación fue demostrar la importancia del rol del arquitecto de información en el proceso de desarrollo de software, para la cual se analizan teóricamente los elementos que identifican el proceso de AI y se describen las Metodologías Rup y XP en cuanto a roles.

Gráfica:

Reflexión:

En un entorno cada vez más dominado por la automatización y el uso de la inteligencia artificial, el rol del arquitecto de información no solo sigue siendo relevante, sino que se ha intensificado. Este profesional no solo se encarga de estructurar la organización del conocimiento y la información dentro del sistema, sino que también es esencial para garantizar que el software sea escalable, sostenible y funcional a lo largo del tiempo.

Cita APA – Bibliografía:

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.

Artículo 10

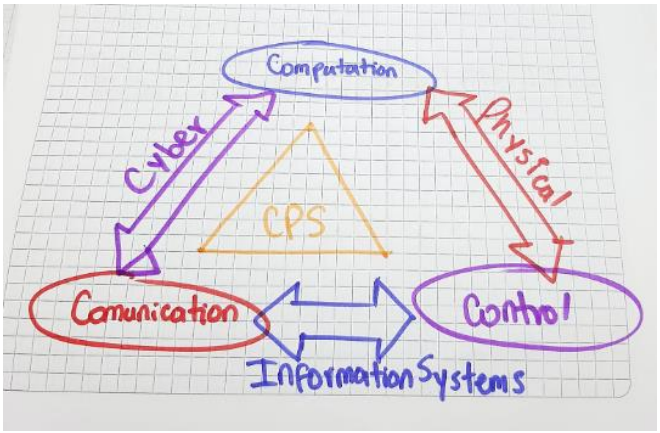
Título:

Arquitectura de Software para los actuales sistemas ciberfísicos.

Resumen:

La próxima generación de sistema ciber-físicos, plantea grandes desafíos en el diseño de software. No se trata sólo de diseñar un sistema entorno a los plazos de ejecución, lo más importante es maximizar la utilización de recursos. En el artículo se proponía un sistema base para la arquitectura de software en el que los servicios se puedan diseñar e implementar y componer fácilmente de acuerdo con la demanda.

Gráfica:



Reflexión:

Los CPS, al integrar componentes computacionales y físicos de manera estrecha, no solo requieren sistemas de software robustos, sino que deben maximizar la eficiencia en el uso de los recursos disponibles. Esta tarea se complica aún más cuando se deben tener en cuenta factores como la latencia, la confiabilidad, la seguridad y la escalabilidad.

Cita APA- Bibliografía:

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

Artículo 11

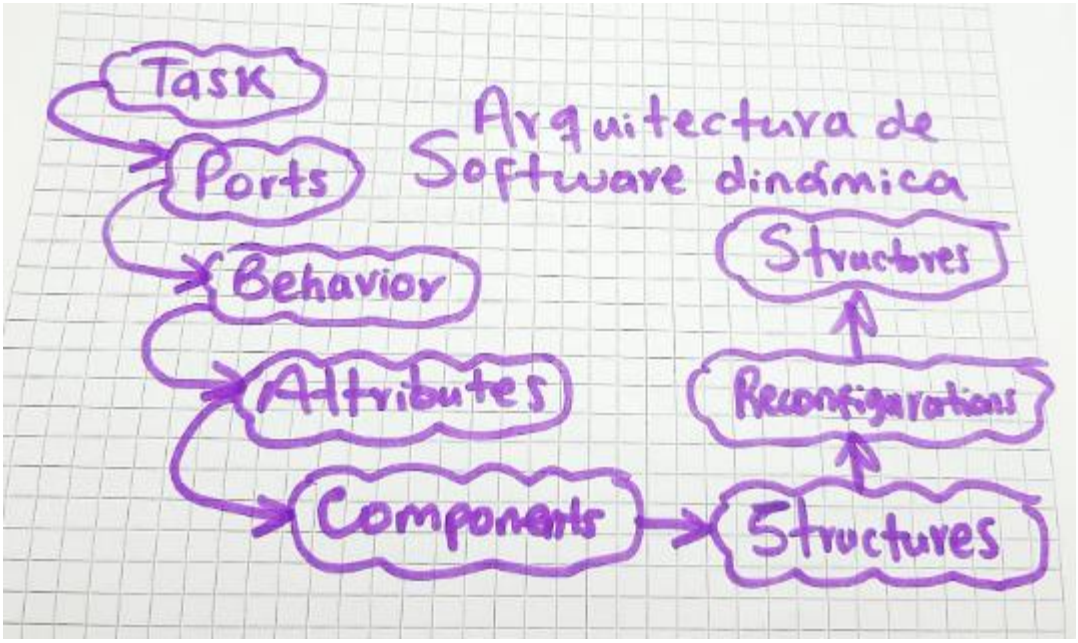
Título:

Arquitectura de software dinámica basada en la reflexión.

Resumen:

La tesis sobre el artículo, se centraba en el campo de la arquitectura de software, una rama de la Ingeniería de software dedicada al estudio de la estructura de los sistemas software complejos. Se trata y estudia concretamente uno de los problemas pendientes del campo: la especificación y descripción de arquitecturas de software dinámicas, es decir, aquellas cuya estructura puede variar. Con esto se elabora de manera informal un modelo reflexivo de descripción arquitectónica de software.

Gráfica:



Reflexión:

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 software complejos**. Este campo se encarga de analizar cómo se organizan y se interrelacionan los componentes dentro de un sistema para garantizar su efectividad, escalabilidad, y flexibilidad.

Cita APA- Bibliografía:

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

Artículo 12

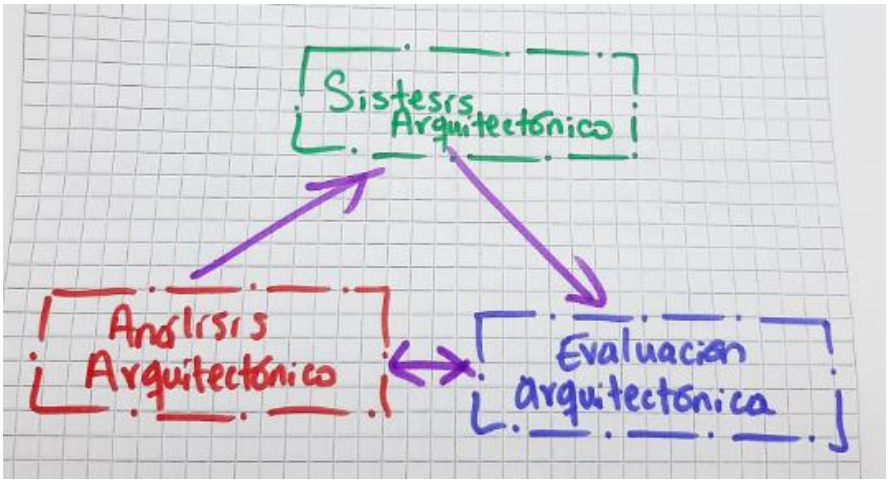
Título:

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

Resumen:

El objetivo general de la tesis estipulada en el artículo, estaba directamente vinculada a brindar soporte al arquitecto de software durante el diseño arquitectónico. Partiendo de la hipótesis: los arquitectos de software reutilizan soluciones conocidas en nuevos diseños, se identifica la necesidad de contar con una herramienta, tanto conceptual como computacional, que pueda colaborar con los arquitectos de software.

Gráfica:



Reflexión:

La **reutilización de soluciones conocidas**. Esta práctica, que consiste en aplicar patrones, diseños o componentes previamente probados y validados, es una estrategia fundamental para mejorar la eficiencia, reducir los costos de desarrollo y aumentar la calidad del software. La hipótesis planteada — 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.

Cita APA- Bibliografía:

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).

Artículo 13

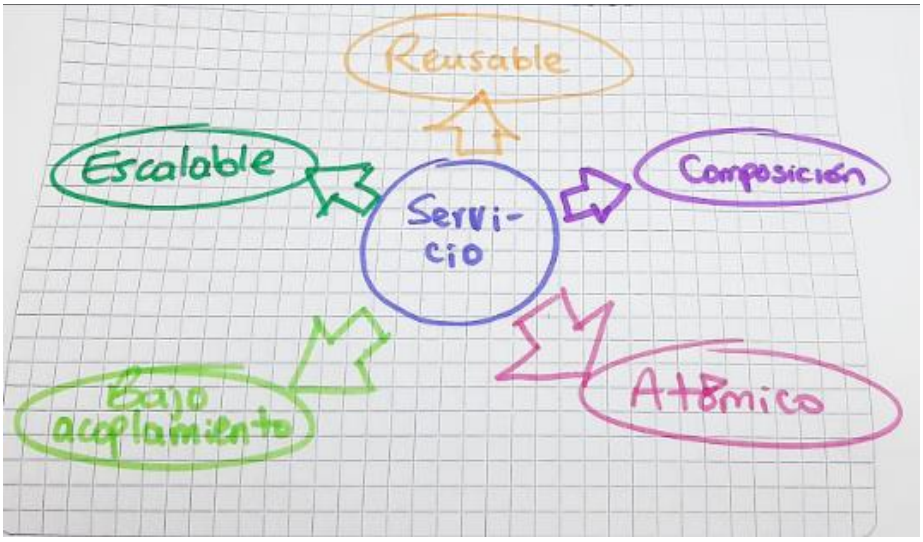
Título:

Arquitectura de software. Arquitectura orientadas a servicios.

Resumen:

Dentro del proceso de desarrollo de software de un sistema de los temas importantes a tratar es: la arquitectura de software, es la parte de la ingeniería de software que se dedica al estudio, análisis y descripción para formalizar el esquema global de un sistema; poniendo énfasis en el estudio de las interacciones entre sus elementos básicos, denominados componentes. En el artículo se realizó un estudio general de la arquitectura de software haciendo énfasis en la arquitectura orientada a servicios.

Gráfica:



Reflexión:

La **arquitectura de software** es un componente fundamental dentro del proceso de desarrollo de software, ya que establece las bases para la estructura general del sistema. Su objetivo es organizar los diversos elementos que conforman el sistema de forma eficiente, asegurando que interactúen correctamente entre sí y con otros sistemas externos. Se encarga de **describir, analizar y formalizar el esquema global** del sistema, lo que incluye la distribución de tareas, la asignación de responsabilidades y las relaciones entre los diferentes **componentes** del sistema.

Cita APA. Bibliografías:

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

Artículo 14:

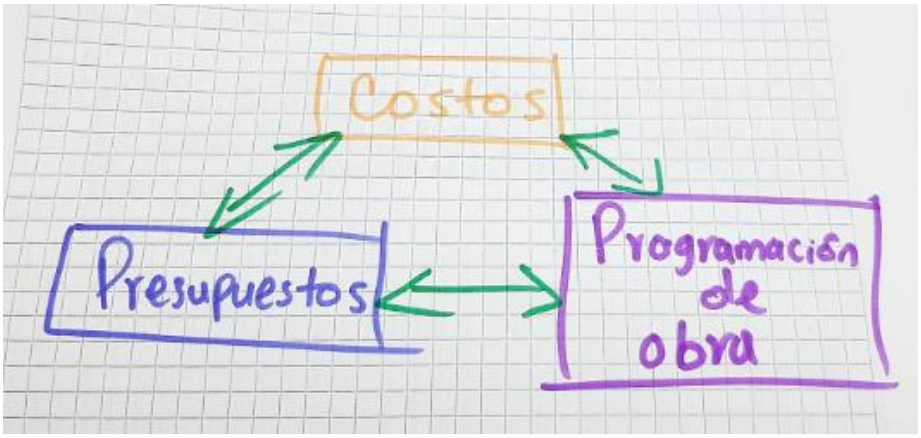
Título:

Arquitectura de Software para el desarrollo de herramienta tecnológica de costos, presupuestos y programación de obra.

Resumen:

Durante todo el artículo trataron temas relacionados a la arquitectura de software, como los casos de uso, diagramas de flujo del software, diagramas de despliegue, diagramas de paquetes del software. Por último, nombraron todo sobre la cronología, metodología de desarrollo de costos, presupuestos y programación de obra que se debe llevar a cabo. En calidad, establece listas, establece valores de los costos indirectos, calcular el costo total del proyecto y realizar la programación de obra.

Gráfica:



Reflexión:

La **arquitectura de software** es la base sobre la cual se construye el sistema, y como tal, debe ser cuidadosamente diseñada y documentada. Los **diagramas de flujo**, los **diagramas de despliegue**, los **casos de uso** y los **diagramas de paquetes** permiten representar de forma visual y estructurada cómo interactúan los diferentes componentes del sistema. Estos artefactos son fundamentales para garantizar que todos los elementos del software estén bien definidos y que sus interacciones sean claras.

Cita APA- Bibliografía:

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.

Artículo 15

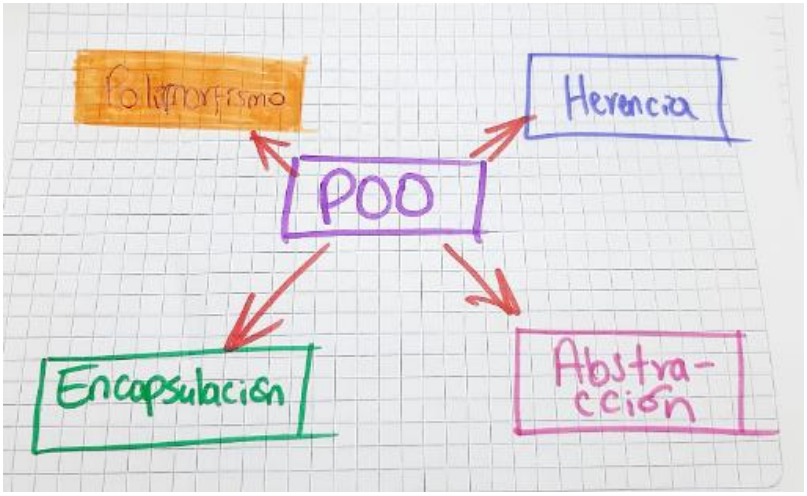
Título:

Arquitectura de software con programación orientada a objetos.

Resumen:

El objetivo del artículo fue describir la arquitectura de software con programación orientada a objetos a partir de la revisión de fuente de documentales actualizadas. Se encontró que la POO, agrupa un conjunto de técnicas que permiten desarrollar, y mantener mucho más fácilmente, programas de una gran complejidad. La evolución de la arquitectura de software, escala cada vez más posiciones superiores, que permiten a los profesionales resolver diversos problemas.

Gráfica:



Reflexión:

La **arquitectura de software** con **programación orientada a objetos (POO)** ha demostrado ser una de las metodologías más poderosas y efectivas para el desarrollo de software, especialmente en el contexto de 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 facilitan no solo el desarrollo, sino también el **mantenimiento y evolución** de programas con un alto grado de complejidad.

Cita APA- Bibliografía:

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

Artículo 16

Título:

La importancia de la Arquitectura de software

Resumen:

El software es un conjunto de programas de cómputo, procedimientos, estructura de datos, reglas documentación y datos asociados que forman parte de las operaciones de un sistema de computación. Se necesita tener un panorama completo del sistema, como el ciclo de vida de la arquitectura, en donde hayan requisitos, un diseño, documentación, evaluación e implementación; en general, donde haya una buena práctica de la arquitectura.

Gráfica:

Reflexión:

La definición de **software** como un conjunto de programas, procedimientos, estructuras de datos, reglas, documentación y datos asociados resalta la **complejidad integral** de lo que implica crear y mantener sistemas de computación. Es crucial reconocer que, más allá de escribir código, el software es un **sistema holístico** que requiere un enfoque organizado y sistemático para garantizar su funcionalidad, eficiencia y calidad.

Cita APA- Bibliografía:

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

Artículo 17

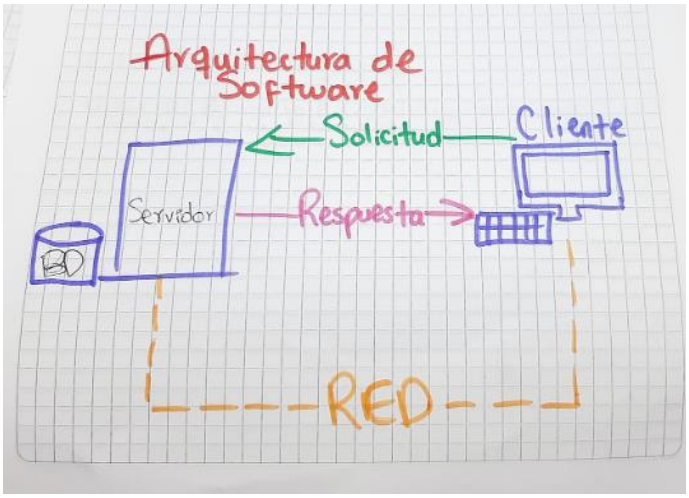
Título:

Arquitectura de software

Resumen:

La arquitectura de software permite evaluar la solución de un software desde las primeras fases de su desarrollo, apartando numerosos beneficios a cualquier proyecto de software que se realice. Permite la evaluación de la solución desde fases muy tempranas de desarrollo. Determina la viabilidad de un proyecto, si tiene grandes y complejos sistemas.

Grafica:



Reflexión:

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

Cita APA- Bibliografía:

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

Artículo 18

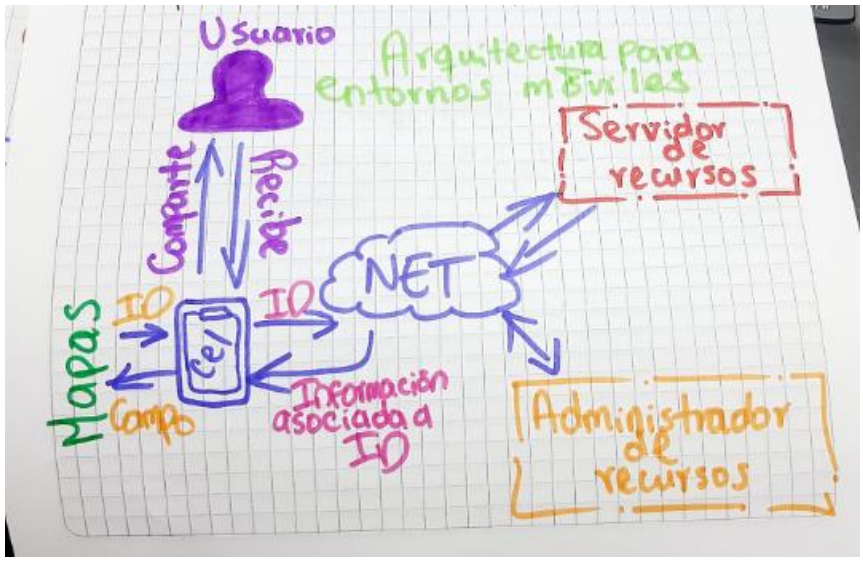
Título:

Arquitectura de software para entornos móviles

Resumen:

La evaluación de los sistemas operativos está causando un gran impacto, porque cada vez son más estables y robustos, lo que permite a los desarrolladores de software, crear aplicaciones móviles de mayor tamaño y complejidad. El objetivo del artículo fue definir una solución arquitectónica móvil que compartiera algunos de los principios más reconocidos de la arquitectura de software em general, y de esa forma, ayudar a estandarizar y adaptar metodologías, métodos, procesos y enfoques.

Gráfica



Reflexión:

La **evaluación de los sistemas operativos móviles** y su evolución hacia soluciones más estables y robustas ha tenido un impacto significativo en la capacidad de los desarrolladores para crear **aplicaciones móviles de mayor tamaño y complejidad**. Este avance no solo ha mejorado el rendimiento y la fiabilidad de las aplicaciones, sino que también ha proporcionado una **base más sólida** sobre la cual construir software innovador y de alta calidad.

Cita APA- Bibliografía:

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.

Artículo 19

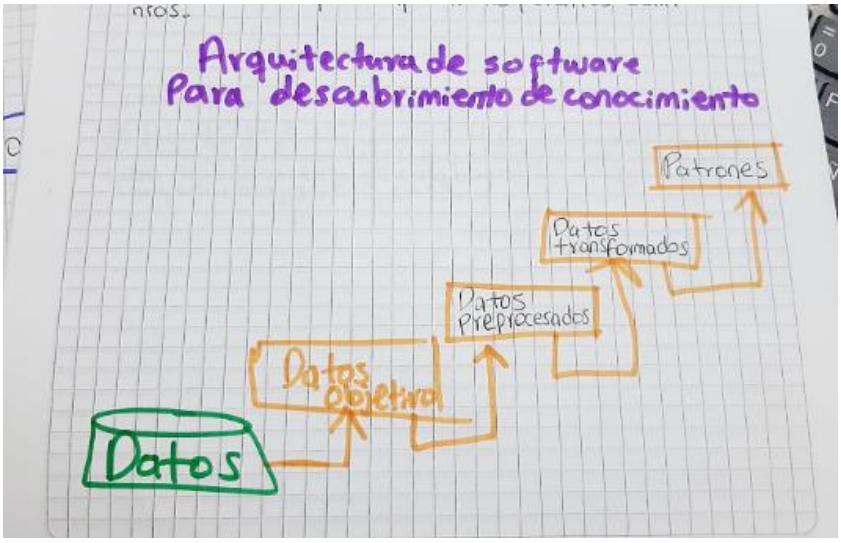
Título:

Arquitectura de software para descubrimiento de conocimiento

Resumen:

La mayoría de las organizaciones cuentan con sistemas de gestión de bases de datos. Se puede descubrir conocimientos en grandes volúmenes de datos que se encuentran almacenados, ya sea en archivos o en bases de datos. Es un proceso interactivo e iterativo que implica integrar en una herramienta interrogaciones, análisis y métodos de visualización que le facilitan al usuario. El proceso de descubrimiento de conocimiento se puede aplicar en diferentes dominios.

Gráfica



Reflexión:

El proceso de descubrimiento de conocimiento en bases de datos ha ganado una 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 proporcionar un valor significativo para la toma de decisiones, la optimización de procesos y la innovación empresarial.

Cita APA- Bibliografía:

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.

Artículo 20

Título:

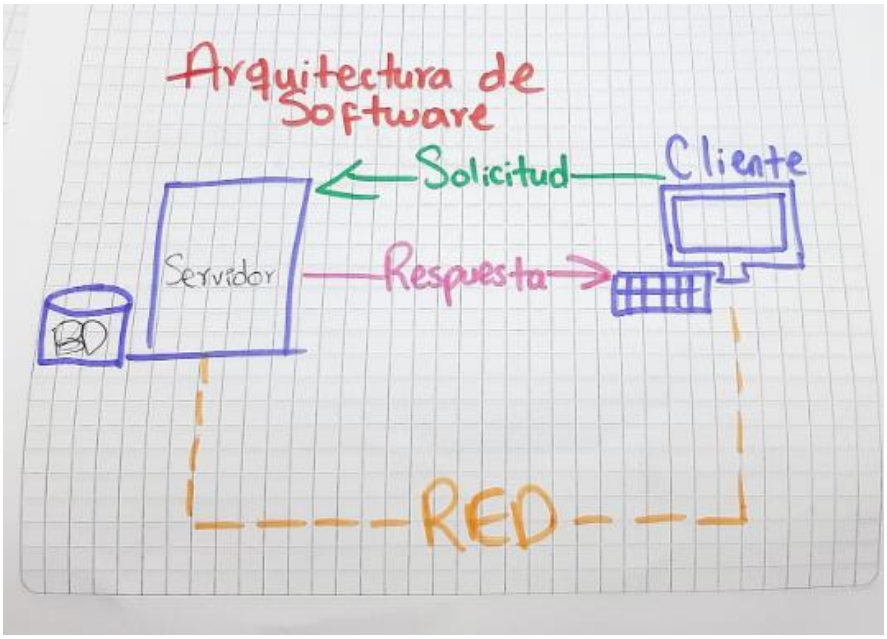
Diseño de un mundo virtual para la enseñanza de arquitectura de software

Resumen:

El objetivo principal del proyecto hablado en el artículo fue crear un mundo virtual en el que se enseñe Arquitectura de Software a nivel universitario. Algo importante para lograr este fin fue la investigación sobre las tecnologías, lo que hizo que dicha idea fuera viable.

Se estudiaron herramientas que pudieron ayudar a la consecución del proyecto y a la implantación del mismo, contando con las diferentes posibilidades que se plantearon. Definir los pasos que se deberían realizar para la construcción del sistema.

Gráfica:



Reflexión:

La creación de un mundo virtual para enseñar Arquitectura de Software a nivel universitario es una iniciativa innovadora que destaca la importancia de integrar nuevas tecnologías en el proceso educativo. La idea de utilizar un entorno virtual para enseñar conceptos complejos como la arquitectura de software no solo hace que el aprendizaje sea 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.

Cita APA- Bibliografía:

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

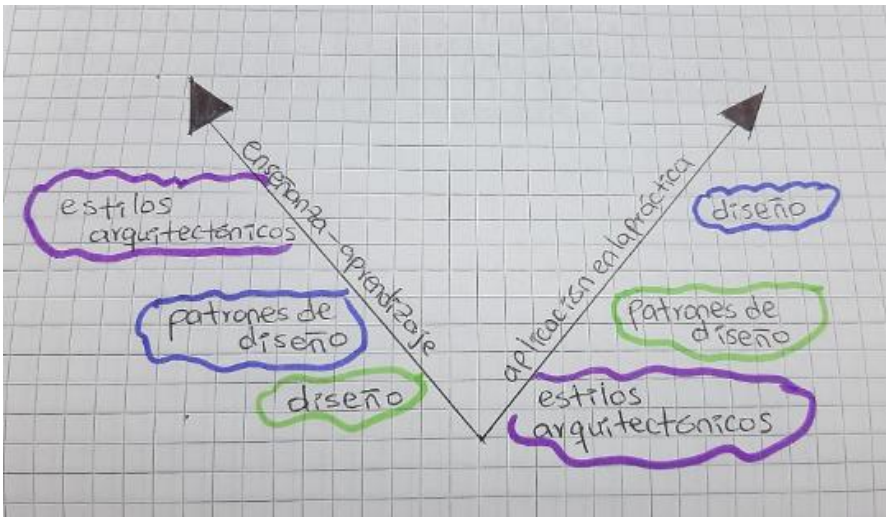
Artículo 21

Título:

Una teoría para el diseño de software.

Resumen:
El resultado de diseñar un sistema es una de las cuatro descripciones fundamentales que debe realizar un ingeniero de software. A esta descripción se la conoce como diseño de software, arquitectura de software o estructura de software. Es conveniente dedicar tiempo y esfuerzo en diseñar un sistema de software porque es más barato desarrollarlo y mantener, que hacerlo como se lo hace habitualmente. El mantenimiento consiste esencialmente en cambiar, agregar o eliminar líneas de código al software.

Gráfica:



Reflexión:

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 la 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.

Cita APA- Bibliografía:

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

Artículo 22

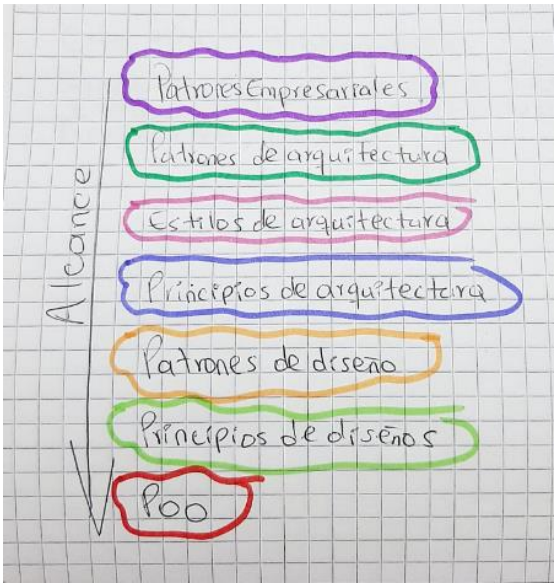
Título:

Definición de arquitectura para una línea de producto de software

Resumen:

El artículo el cual trataba sobre el arte en la definición de arquitecturas en las líneas de producto de software, por lo que en él se trataron diversos conceptos de Arquitectura de software encontrados en la literatura revisada. El modelado de arquitectura es el esfuerzo de capturar y documentar las decisiones de diseño que conforman la arquitectura del sistema. Una línea de producto es un conjunto de sistemas que comparten un conjunto de características para satisfacer las necesidades específicas de un segmento de mercado particular.

Gráfica:



Reflexión:

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.

Cita APA- Bibliografía: Cita APA - Bibliografía:

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

Artículo 23:

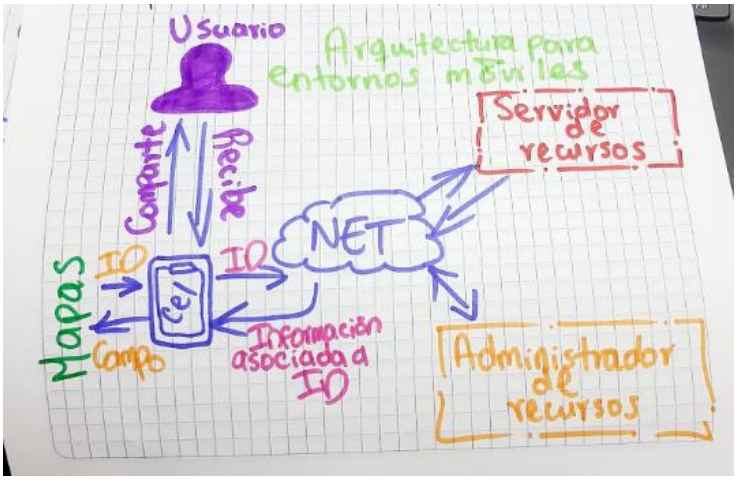
Título:

SERVICIOS WEB EN TELEFONÍA CELULAR

Resumen:

La evolución de la telefonía móvil ha permitido el desarrollo de estándares con el fin de facilitar la adquisición y conocimiento de servicios Web a los usuarios de móviles. La llegada de nuevas tecnologías como SOA (Service Orient to Architecture) abre un amplio campo de posibilidades para el desarrollo de diversas aplicaciones cuyo propósito es facilitar el acceso a contenido informativo e interactivo.

Gráfica:



Reflexión:

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.

Cita APA:

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

Artículo 24:

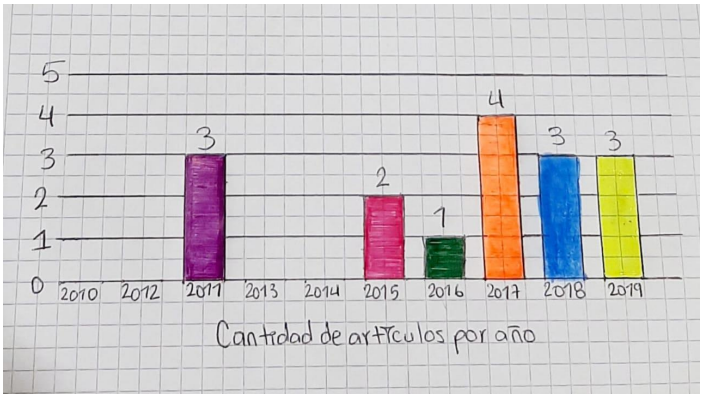
Título:

Protección de Datos Personales en Ecuador y Colombia: Principios, Ética y Desafíos Actuales

Resumen:

Se analizó en profundidad el concepto de protección de datos y del uso de los datos personales en países como Ecuador y Colombia donde se hace referencia a la forma de tratamiento de datos donde se examina por categorías y se especifica los principios que se deben aplicar para realizar el proceso de tratamiento de datos los cuales son: legalidad, libertad, veracidad, transparencia, acceso y circulación restringida, seguridad y confidencialidad. Por otro lado, también manifiesta al perfil ético que tiene el responsable para no cometer actos ilícitos con la información entregada y examina los tratamientos que impliquen el uso de datos biométricos.

Gráfica:



Reflexión:

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 un contexto digital y globalizado. 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.

Cita APA:

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.

Artículo 25:

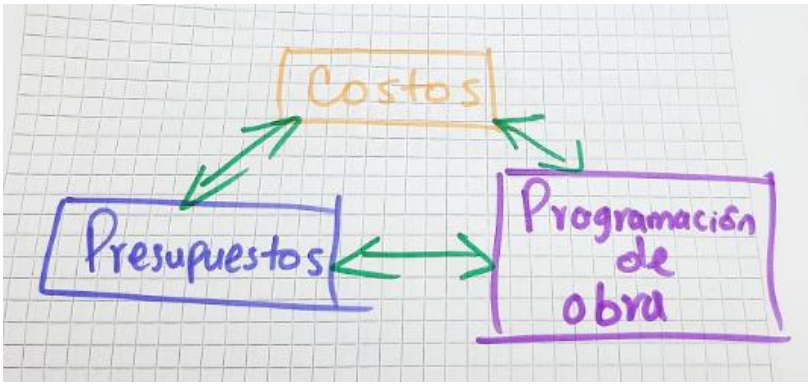
Título:

Criterios de evaluación de plataformas de desarrollo de aplicaciones empresariales para ambientes web

Resumen:

Las aplicaciones web han estado tomando fuerza en los últimos años, esto debido a la practicidad de las mismas, entre otras razones, por la independencia del sistema operativo, así como a la facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software a miles de usuarios potenciales y son accesibles desde cualquier lugar del mundo gracias a la red de redes, Internet. Los mecanismos de desarrollo de aplicaciones Web, recogen elementos comunes al desarrollo de aplicaciones empresariales, pero tienen características propias en análisis, diseño, e implementación. Estos elementos serán independientes del estilo arquitectónico que se decida implementar y también de la arquitectura de software.

Gráfica:



Reflexión:

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.

Cita APA:

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

Artículo 26:

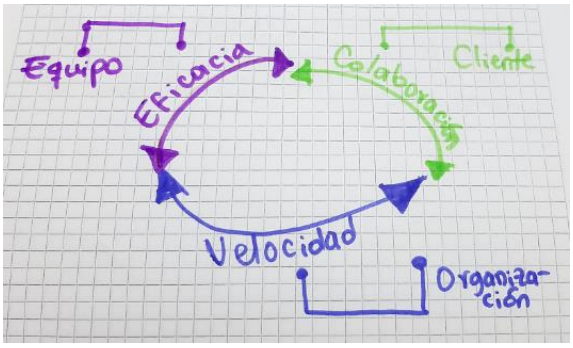
Título:

Análisis comparativo de Patrones de Diseño de Software

Resumen:

Los patrones de diseño brindan soluciones a problemas que se presentan durante el desarrollo de software, evitan duplicaciones de código y facilitan su reutilización. En el presente artículo se detallan la estructura, componentes, ventajas y desventajas de los patrones de diseño: Template Method, Model-View-Controller, Model-View-Presenter, Model Front Controller y Model-View-View-Model MVVM. La investigación se realizó a través de una revisión bibliográfica en bases de datos científicas y consecuentemente se determinaron las métricas que permitieron comparar los patrones en estudio. Mediante el análisis comparativo de métricas y parámetros entre los patrones se establece que no existe un patrón superior a nivel general, pues cada patrón tiene su propósito definido y el desarrollador de software es quien debe identificar cuando un patrón se adapta mejor a la solución que desea desarrollar.

Gráfica:



Reflexión:

Es interesante cómo los patrones de diseño funcionan como 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.

Cita APA:

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.

Artículo 27:

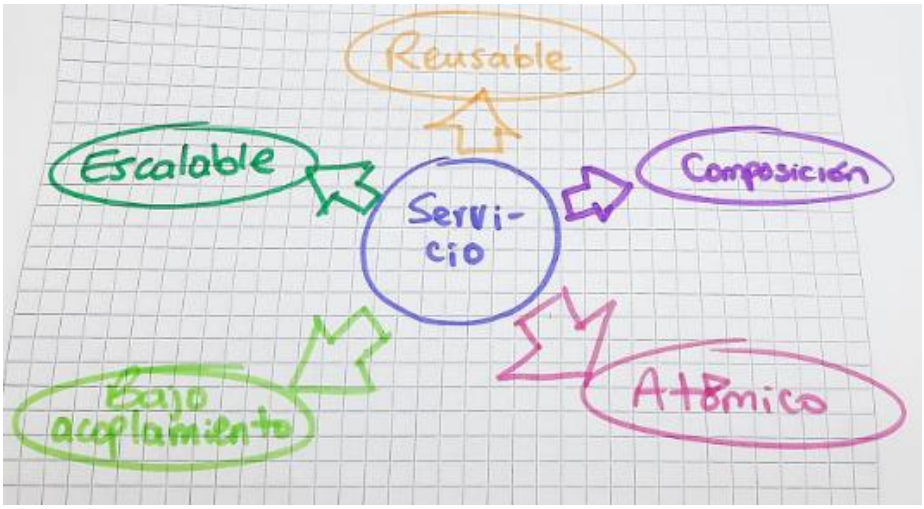
Título:

Arquitectura de software, esquemas y servicios

Resumen:

Cuando en la industria de software los productos tienen requerimientos cada vez más complejos y dinámicos, y los tiempos para desarrollarlos son cada vez menores; la reutilización y el bajo acoplamiento entre los componentes cobran vital importancia. En el artículo de investigación, partiendo de las definiciones de arquitectura de software y esquema, se tratan las características del paradigma de arquitectura orientada a servicios y se exponen algunos elementos significativos que muestran cómo los servicios son la evolución natural de los componentes de software. También se comentaban algunas cuestiones a tener en cuenta a la hora de diseñar orientado a servicios.

Gráfica:



Reflexión:

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.

Cita APA:

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

Artículo 28:

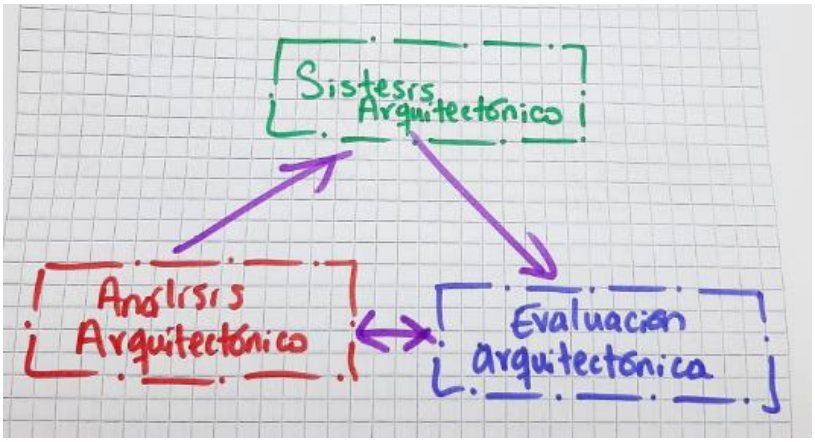
Título:

Esquema basado en UML para representaciones de arquitectura de software

Resumen:

En los últimos años se ha llevado a cabo una importante cantidad de investigaciones en el campo de la arquitectura de software. El objetivo de muchos de estos estudios es encontrar un sistema de representación capaz de ir más allá de la informalidad del diagrama tradicional de "cajas y líneas", pero manteniendo un nivel de complejidad bajo, de modo que pueda utilizarse como herramienta de comunicación entre todos los interesados en un proyecto de software.

Gráfica:



Reflexión:

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.

Cita APA:

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

Artículo 29:

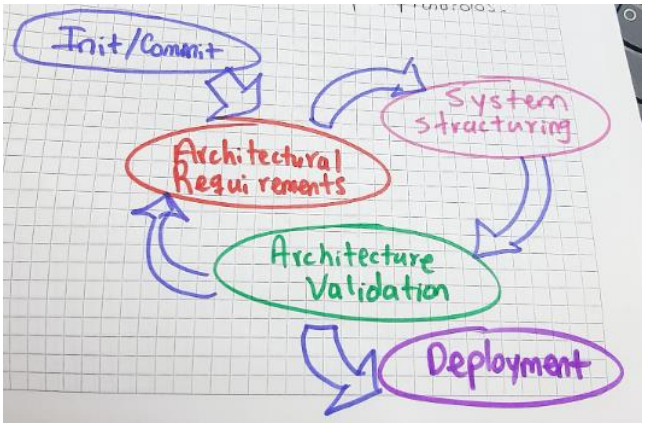
Título:

Arquitectura de Software de Referencia para Objetos Inteligentes en Internet de las Cosas

Resumen:

La evolución de los sistemas embebidos, que tuvo lugar junto al bajo costo y ubicuidad de internet, ha tenido como resultado el paradigma Internet de las Cosas. El objetivo de este paradigma es convertir los objetos que nos rodean en objetos inteligentes de forma tal que, comunicándose a través de Internet, puedan percibir lo que sucede en su entorno y poder reaccionar frente a ello. Se han llevado a cabo numerosos trabajos en materia de estandarización para este paradigma como, por ejemplo, protocolos de comunicación, topologías de red y arquitecturas software de alto nivel que consideran, por sobre todas las cosas, cómo integrar todas las piezas de una solución de Internet de las Cosas en un solo sistema.

Gráfica:



Reflexión:

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.

Cita APA:

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.

Artículo 30:

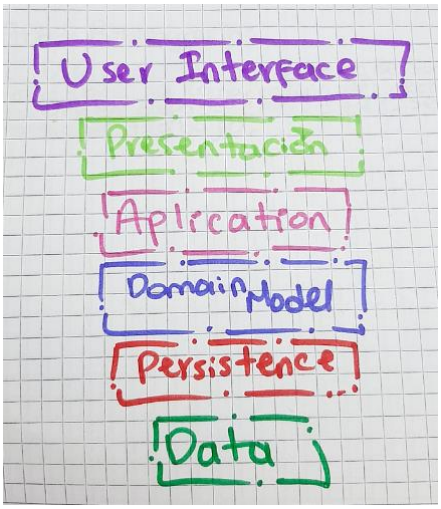
Título:

Arquitectura software de sistemas abiertos

Resumen:

El aumento de la complejidad de los sistemas software ha puesto de manifiesto la importancia que tiene la Arquitectura del Software en todo el proceso de desarrollo y mantenimiento del mismo. Frente a una visión tradicional centrada fundamentalmente en lo que se concoce como diseño arquitectónico, enmarcado dentro del campo más amplio de la Ingeniería del Software, los trabajos más recientes están orientados a considerar la Arquitectura del Software como un nuevo campo de interés por sí mismo. En este campo, los principales objetivos siguen siendo la especificación y diseño de los aspectos estructurales del software, pero haciendo un mayor énfasis en la especificación de las interrelaciones entre componentes y en el diseño de lenguajes de descripción de arquitecturas bien definidos.

Gráfica:



Reflexión:

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.

Cita APA:

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).

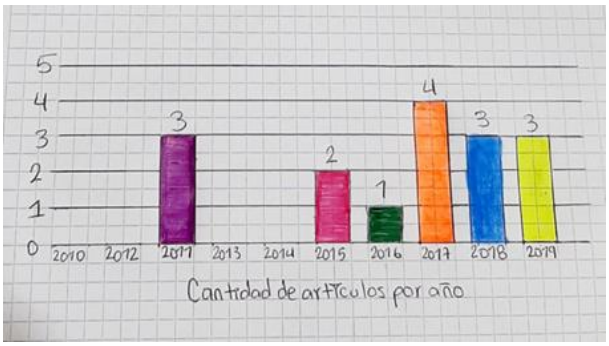
Artículo 31:

Título: Visión de las competencias en arquitectura de software integrando las perspectivas de la industria y la academia

Resumen:

La formación de un arquitecto de software es una labor compleja que requiere de una mezcla de experiencia y conocimiento especializado que es difícil lograr en el contexto universitario. Este artículo busca determinar las competencias mínimas que debe lograr un arquitecto de software cubriendo la expectativa de la industria, así como el contexto formativo de las universidades e instituciones de educación superior. Para identificar y documentar estas competencias, se realizó un ciclo de investigación-acción, en el cual se diseñó un estudio basado en encuestas y talleres en el que participaron ingenieros de software de la industria y profesores universitarios que imparten cursos relacionados con el diseño y evaluación de la arquitectura.

Gráfica:



Reflexión:

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.

Cita APA:

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.

Artículo 32:

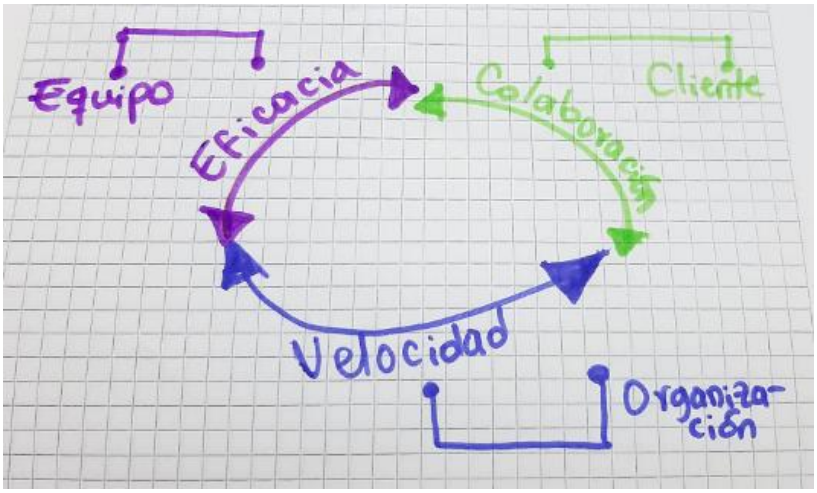
Título:

Metodologías Ágiles en el Desarrollo de Software

Resumen:

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte tenemos aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en otros muchos. Una posible mejora es incluir en los procesos de desarrollo más actividades, más artefactos y más restricciones, basándose en los puntos débiles detectados.

Gráfica:



Reflexión:

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 esencia.

Cita APA:

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

Artículo 33:

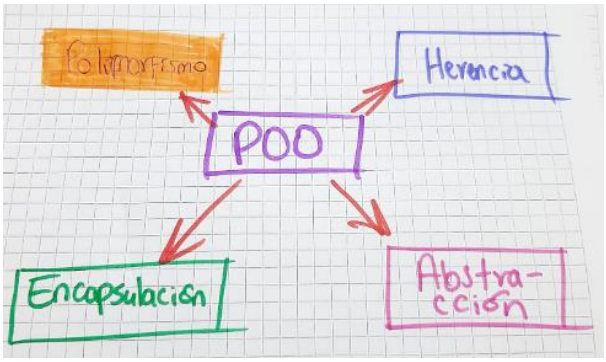
Título:

RADS: una herramienta para reutilizar estrategias en diseños de arquitecturas de software

Resumen:

El diseño de arquitecturas de software es un proceso altamente creativo que aún no ha sido estandarizado, por lo que las actividades llevadas a cabo para construir la arquitectura de un sistema son aquellas que los arquitectos involucrados consideran convenientes y pertinentes según el método de diseño que utilicen, su experiencia, conocimientos y habilidades personales. La reutilización es una práctica habitual dentro de dicha actividad. Sin embargo, no existen herramientas que asistan a los arquitectos para llevarla a cabo.

Gráfica:



Reflexión:

La reflexión sobre el diseño de arquitecturas de software como un proceso creativo, y la falta de herramientas específicas para la reutilización, revela varia. 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.

Cita APA:

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).

Artículo 34:

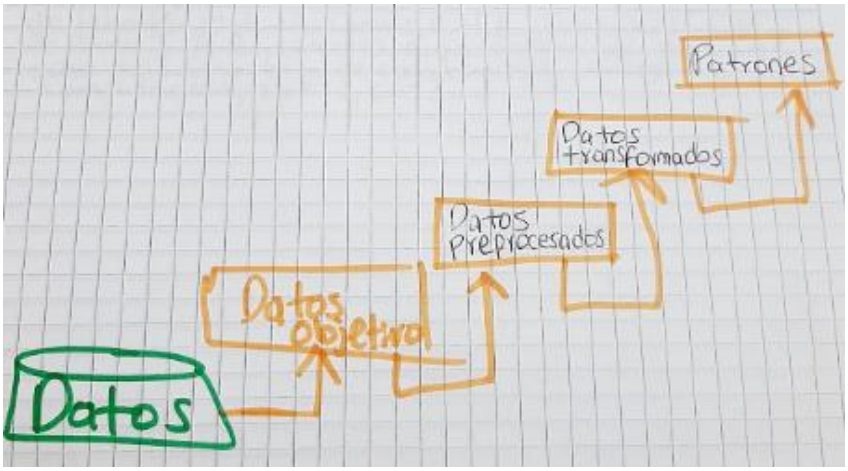
Título:

Evolución de las Metodologías y Modelos utilizados en el Desarrollo de Software

Resumen:

las metodologías de Desarrollo de Software (DS.) han experimentado un proceso histórico y evolutivo que inicia en los años 40 con la aparición de las primeras computadoras, entonces no se contaban con parámetros ni estándares, el DS. Era prácticamente empírico y artesanal lo que llevó a que una buena parte de los proyectos fallaran en cubrir las exceptivas de los usuarios, así como en entregas extemporáneas y presupuestos excedidos, sobreviniendo la "crisis del Software" la respuesta para superarla fue la adopción de modelos y metodologías clásicas que progresivamente fueron incorporando estándares, controles y formalidades al DS. En un afán que llegó a ser definido como "triángulo de hierro."

Gráfica:



Reflexión:

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 cumplan.

Cita APA:

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