Artículos de arquitectura de software y patrones de diseño

Angie Lizeth Trujillo Gonzalez

Trabajo presentando para el curso de elaboración de artículos científicos en actividades de investigación.

Instructor

Jesús Ariel González Bonilla
Ingeniero de sistemas

Servicio nacional de aprendizaje (SENA)
Análisis y desarrollo de software
2024

Introducción.

La arquitectura de software es la estructura fundamental de un sistema de software, que incluye sus componentes, sus relaciones y cómo interactúan entre sí. Define los principios y directrices que guían el diseño, desarrollo y evolución del sistema, asegurando que sea escalable, mantenible y eficiente. Una buena arquitectura proporciona una base sólida para cumplir con los requisitos funcionales y no funcionales, como la seguridad, el rendimiento y la flexibilidad.

Por otro lado, los patrones de diseño son soluciones probadas y reutilizables para problemas comunes en el diseño de software. Actúan como una especie de "receta" que ayuda a los desarrolladores a abordar desafíos específicos en el desarrollo de sistemas. Los patrones no son implementaciones directas, sino guías que pueden adaptarse a las necesidades específicas del proyecto. Ejemplos clásicos incluyen el patrón Singleton, el Observador, el Modelo-Vista-Controlador (MVC) y el Fachada, entre otros.

En conjunto, la arquitectura de software y los patrones de diseño ayudan a los equipos a crear sistemas más robustos, organizados y fáciles de mantener, fomentando una mejor comunicación y colaboración entre los desarrolladores.

BIOGRAFÍA

Mi nombre es Angie Trujillo, soy desarrolladora de software. Cuento con una sólida formación en tecnología adquirida en el SENA. A la edad de 19 años, he tenido la oportunidad de adentrarme en el ámbito del desarrollo tecnológico, elaborando proyectos innovadores y funcionales que convierten ideas en realidades digitales.

Desde el inicio, he sido motivada por el anhelo de adquirir conocimiento y enfrentar desafíos, lo que me ha destacado por mi habilidad para solucionar dificultades y aportar valor en cada proyecto en el que me involucro. Mi compromiso con la excelencia y mi entusiasmo por la tecnología me motivan a continuar expandiendo mis horizontes y a explorar nuevas oportunidades en este apasionante ámbito.

Mi objetivo es continuar desarrollando una trayectoria profesional que motive a otros jóvenes a confiar en sus capacidades y a animarse a marcar su presencia en el campo de la tecnología.

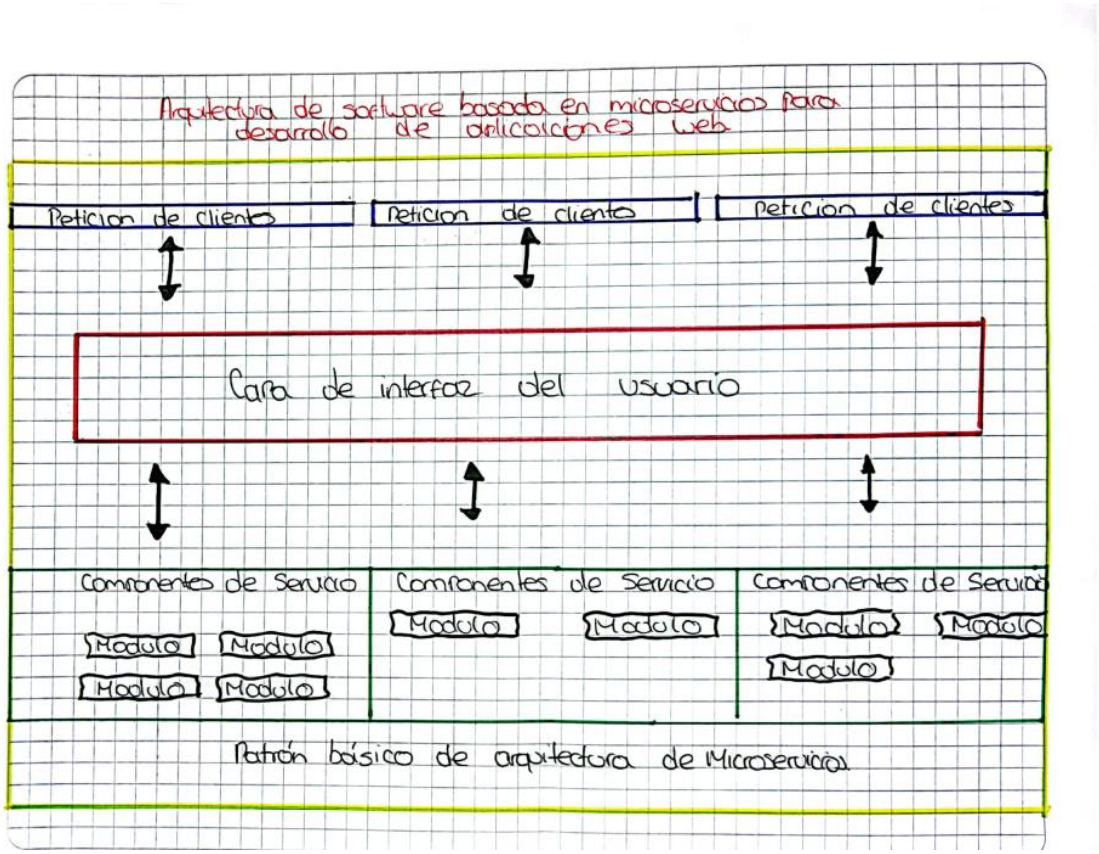
Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web.

Resumen

El desarrollo de software en la Coordinación General de Tecnologías de la Información y Comunicación (CGTIC) de la Asamblea Nacional del Ecuador (ANE) se ha basado en una arquitectura monolítica. Este enfoque empaqueta toda la funcionalidad en una única unidad ejecutable, siguiendo las tendencias impuestas por el lenguaje de programación utilizado y la experiencia del área de desarrollo. Si bien este modelo ha sido funcional, ha generado desafíos significativos en términos de mantenimiento, escalabilidad y entrega de software, especialmente a medida que las necesidades y los sistemas han evolucionado.

Reflexión

El uso de arquitecturas monolíticas presenta problemas en sistemas complejos, dificultando el mantenimiento y la escalabilidad. Esto resalta la necesidad de adoptar arquitecturas modernas, como los microservicios, que dividen aplicaciones en componentes independientes. Esta transición mejora tanto el mantenimiento como la escalabilidad, permitiendo un trabajo más ágil y eficiente para los equipos de desarrollo. La investigación actual puede transformar la forma en que la CGTIC aborda el desarrollo de software, promoviendo soluciones sostenibles.



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

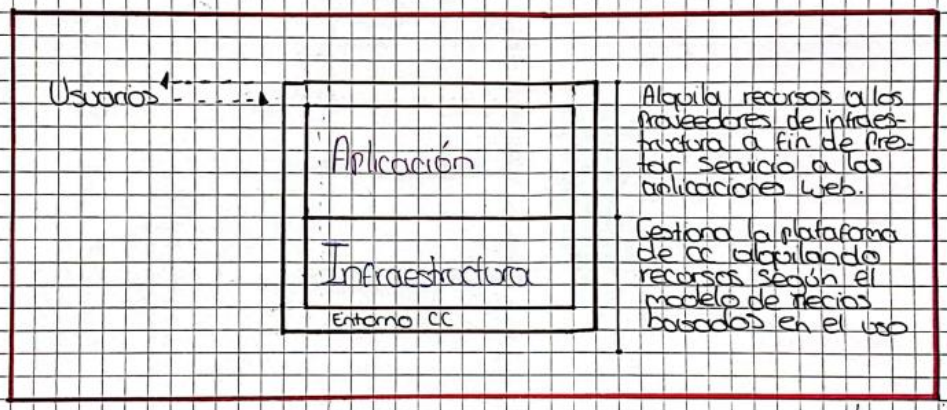
Resumen

Presenta un enfoque integral para el diseño de arquitecturas de software, especialmente para aplicaciones web. Se destaca la importancia de utilizar arquitecturas genéricas que proporcionen una estructura común para resolver problemas específicos, lo que ayuda a los arquitectos a evitar conflictos derivados de la falta de experiencia. El trabajo introduce un metamodelo de componentes arquitectónicos que identifica elementos clave en el diseño, y se complementa con una herramienta gráfica para la instalación de estos componentes. Además, se aborda la verificación de patrones de diseño para asegurar su correcta aplicación, lo que contribuye a la calidad del software. El modelo UML se utiliza para describir diferentes aspectos de la arquitectura, incluyendo la aplicación en la nube y la descomposición en capas. Finalmente, se enfatiza que la experiencia del arquitecto es crucial para seleccionar las instancias adecuadas que cumplan con los requerimientos funcionales y de calidad.

Reflexión

El enfoque presentado en el documento es altamente relevante en el contexto actual de la computación en la nube, donde la complejidad de las aplicaciones web está en constante aumento. La propuesta de un entorno de diseño integral que combina un metamodelo con herramientas de verificación es una contribución valiosa, ya que no solo facilita el trabajo de los arquitectos de software, sino que también promueve la creación de aplicaciones más robustas y de calidad. Además, la atención a la experiencia del arquitecto resalta la necesidad de formación continua en un campo que evoluciona rápidamente. En general, este trabajo puede servir como una guía útil para profesionales y académicos interesados en mejorar sus prácticas de diseño en entornos de computación en la nube.

Modelado y verificación de patrones de diseño de arquitecturas de software para entornos de computación en la nube



Descomposición arquitectónica de alto nivel del diseño de aplicaciones web basadas en entorno C.C

Análisis comparativo de Patrones de Diseño de Software

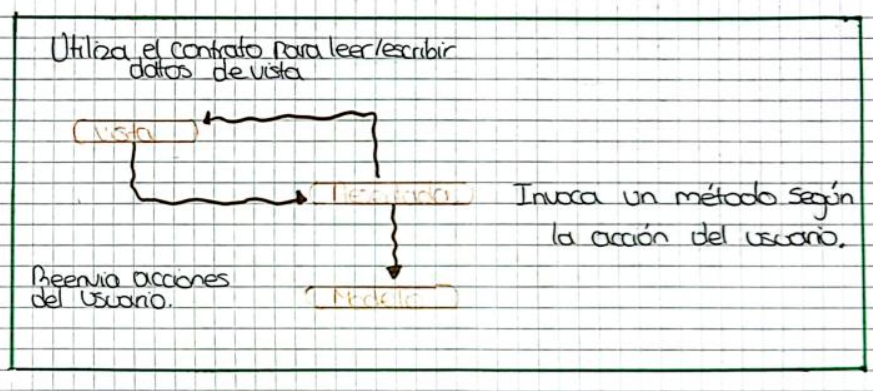
Resumen

En resumen, la evolución de las prácticas de desarrollo de software ha llevado a la implementación de pruebas durante el desarrollo para identificar problemas tempranamente y reducir costos de mantenimiento. Los patrones de diseño son herramientas esenciales que ayudan a estructurar y organizar aplicaciones, resolviendo problemas comunes con soluciones probadas. No son implementaciones directas de código, sino modelos conceptuales que guían a los desarrolladores en la resolución de problemas específicos, mejorando la calidad del software mediante el diseño orientado a objetos. El artículo analiza cinco patrones de diseño destacados: Template Method, Model-View-Controller, Model-View-Presenter, Front Controller y Model-View-ViewModel, destacando sus ventajas y desventajas. La elección del patrón adecuado depende del problema a resolver y del contexto del sistema.

Reflexión

Los patrones de diseño son esenciales para el desarrollo de software moderno, ya que permiten la reutilización de soluciones probadas, reduciendo errores y mejorando la eficiencia. Sin embargo, seleccionar el patrón correcto puede ser un desafío para desarrolladores menos experimentados debido a la amplia variedad disponible. Patrones como MVC, MVP y MVVM ofrecen enfoques específicos para la interacción entre componentes, lo que hace que el software sea más modular y mantenible. Es crucial evaluar cada patrón en función de las necesidades del proyecto, y la documentación es fundamental para educar a los desarrolladores y garantizar un uso efectivo de estas herramientas. En resumen, los patrones de diseño establecen estándares de calidad en la industria y promueven mejores prácticas, siempre y cuando se realice un análisis adecuado y se brinde capacitación adecuada.

Analisis Comparativo de Patrones de diseño de Software



Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web

Resumen

El estudio analiza los patrones de diseño Gang of Four (GOF) presentados en el libro Design Patterns: Elements of Reusable Object-Oriented Software. Estos patrones, clasificados en creacionales, estructurales y de comportamiento, brindan soluciones a problemas comunes en el diseño de software orientado a objetos. Se resalta la importancia de estos patrones como buenas prácticas para mejorar la calidad del software, aunque también se mencionan desafíos como la falta de ejemplos didácticos en el catálogo original. Se sugiere la creación de un catálogo actualizado y didáctico para facilitar la comprensión y aplicación de los patrones GOF en contextos específicos como el desarrollo web. Por último, se propone como trabajo futuro crear un catálogo actualizado y didáctico que facilite la comprensión y la implementación correcta de los patrones GOF en contextos específicos como el desarrollo web.

Reflexión

Los patrones de diseño GOF han sido clave en el desarrollo de software al proporcionar soluciones reutilizables para problemas comunes, pero su alta abstracción dificulta su implementación, especialmente para desarrolladores novatos. Este estudio destaca la importancia de documentación accesible y ejemplos prácticos alineados con los modelos originales. Es esencial seleccionar los patrones con cuidado según el problema y contexto, evitando su uso innecesario para no introducir complejidad adicional. Aunque siguen siendo relevantes, la efectividad de los patrones GOF depende de la experiencia del desarrollador y de herramientas que faciliten su comprensión. Iniciativas como catálogos más didácticos pueden democratizar su uso y mejorar la calidad del software.

Patrones de diseño gof (The going of four) en el contexto de procesos de desarrollo de aplicaciones orientada a la web

NRO	Patron de diseño	Categoría
1	Builder	Creacionales
2	Factory Method	Creacionales
3	Singleton	Creacionales
4	Decorator	Estructurales
5	Fancade	Estructurales
6	Iterator	Comportamiento
7	Strategy	Comportamiento
8	Template Method	Comportamiento

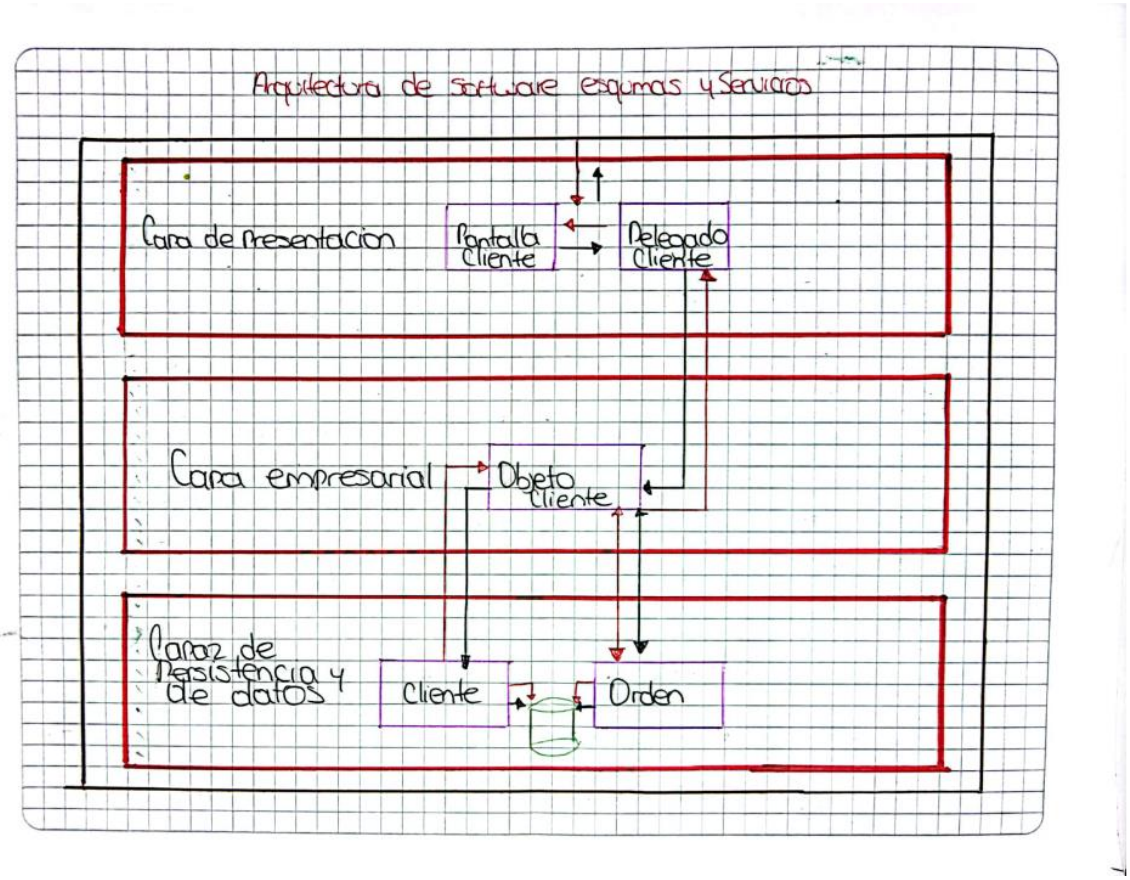
Arquitectura de software, esquemas y servicios.

Resumen

Las aplicaciones empresariales actuales deben cumplir con requisitos que antes eran poco comunes, como la independencia de sistemas operativos y bases de datos, el acceso desde ubicaciones distantes, la interacción con otros sistemas existentes y el manejo de grandes volúmenes de interacciones simultáneas. Esto ha generado una transición de arquitecturas monolíticas y centralizadas a entornos distribuidos y heterogéneos, lo que ha incrementado la complejidad estructural de las aplicaciones. La arquitectura de software ha emergido como una disciplina clave dentro de la ingeniería de software, ya que define las decisiones de diseño fundamentales que estructuran un sistema. Este proceso involucra determinar las partes componentes del sistema, sus interfaces, comportamientos y las relaciones entre ellas, así como su evolución.

Reflexión

La evolución hacia arquitecturas distribuidas y heterogéneas refleja el cambio en las necesidades del mercado empresarial, que exige aplicaciones más flexibles, escalables y accesibles. Esta complejidad, aunque desafiante, ofrece oportunidades para crear soluciones innovadoras que aprovechen tecnologías emergentes. Sin embargo, el desafío para los arquitectos de software radica en gestionar esta complejidad sin sacrificar la eficiencia en el desarrollo, especialmente cuando los recursos, como el tiempo, son limitados. La arquitectura de software se convierte entonces en una disciplina esencial para estructurar aplicaciones que sean sostenibles a largo plazo. Para lograrlo, es crucial que los equipos de desarrollo se enfoquen en principios de diseño sólidos y en una planificación cuidadosa que permita una evolución fluida del sistema. Este enfoque no solo optimiza los recursos disponibles, sino que también asegura que las aplicaciones puedan adaptarse a cambios futuros sin grandes reestructuraciones.



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

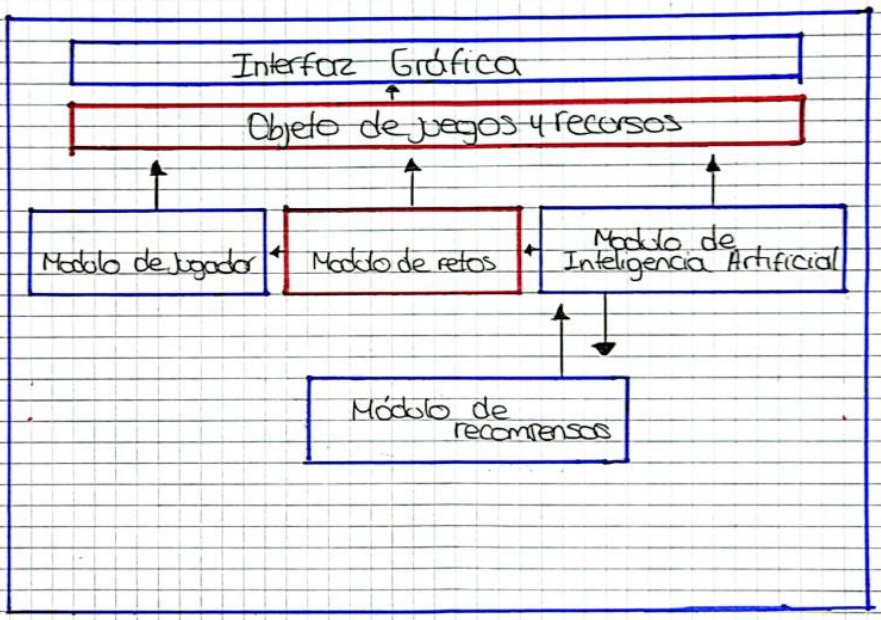
Resumen

El uso de las Tecnologías de Información y Comunicaciones (TIC) ha impulsado el crecimiento de la industria de videojuegos. Entre las disciplinas involucradas en su desarrollo se encuentra la programación, diseño y marketing. Los motores de videojuegos como Unity 3D simplifican y automatizan tareas complejas. Sin embargo, en la UCI se detectaron problemas como la falta de organización y reutilización limitada de componentes en videojuegos creados con Unity 3D. Para resolver esto, se diseñó una arquitectura de software que organiza las características funcionales básicas de los videojuegos. Se validó con un prototipo de videojuego de plataformas, demostrando mejoras en la reutilización y optimización de recursos. La propuesta cumple con atributos de calidad como reusabilidad y extensibilidad. Finalmente, se evaluó la solución a través del método ATAM y pruebas basadas en escenarios.

Reflexión

El desarrollo de videojuegos refleja cómo la tecnología y la creatividad pueden converger para generar soluciones innovadoras. Sin embargo, los retos asociados a la organización y estructuración del proceso evidencian la importancia de contar con herramientas y metodologías sólidas que optimicen los recursos y promuevan la calidad del producto final. El diseño de arquitecturas de software específicas para videojuegos no solo facilita la reutilización de componentes y la eficiencia en el desarrollo, sino que también fomenta la sostenibilidad de proyectos a largo plazo. Este tipo de investigación es fundamental para formar desarrolladores capacitados y para mantener la competitividad de la industria de los videojuegos, garantizando productos que cumplan con las expectativas de los usuarios en un mercado altamente exigente.

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



Arquitectura de Software orientada a la creación de micromundos para la enseñanza y el aprendizaje

Resumen

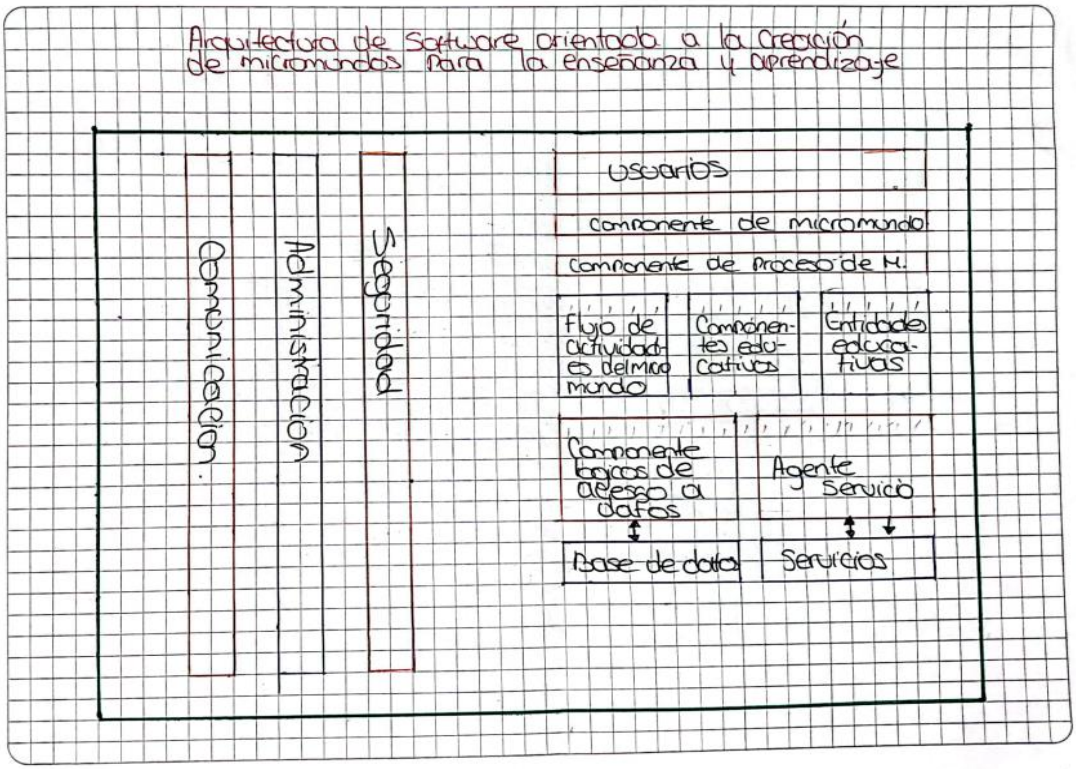
Los procesos de enseñanza-aprendizaje enfrentan diversos desafíos que dificultan el acceso a una educación de calidad. La falta de recursos y el desconocimiento en el manejo de las tecnologías de información limitan tanto a estudiantes como a docentes, mientras que los currículos rígidos restringen la creatividad e impiden una integración efectiva de las TIC. Muchas instituciones educativas no cuentan con recursos tecnológicos suficientes o no los utilizan adecuadamente, desaprovechando el potencial de los estudiantes y afectando su formación. Además, el analfabetismo digital, especialmente en países en desarrollo, genera desigualdades científicas, tecnológicas y educativas con repercusiones sociales y económicas. Ante estos retos, es fundamental implementar herramientas que fomenten el autoaprendizaje, reestructurar los modelos educativos para adaptarlos a las nuevas generaciones y garantizar un uso eficiente de las TIC como complemento en el proceso de enseñanza.

Reflexión

La educación enfrenta desafíos importantes en un mundo cada vez más digitalizado. La falta de integración de las TIC en los procesos de enseñanza limita la capacidad de los estudiantes para adaptarse a las demandas actuales. Es necesario reestructurar los modelos educativos para fomentar habilidades tecnológicas, creatividad y autoaprendizaje, asegurando que todos los actores tengan acceso equitativo a los recursos.

El analfabetismo digital es una barrera crítica en países en desarrollo, perpetuando desigualdades económicas y sociales. Para superarlo, es fundamental invertir en infraestructura tecnológica, capacitar a docentes y estudiantes, y diseñar currículos flexibles que aprovechen el potencial de las herramientas digitales. Solo a través de un cambio sistémico, que priorice la inclusión tecnológica, podremos formar generaciones preparadas para enfrentar los retos del siglo XXI.

Arquitectura de Software orientada a la Creación de micromundos para la enseñanza y aprendizaje



USO DE PATRONES DE DISEÑO DE SOFTWARE: UN CASO PRÁCTICO

Resumen

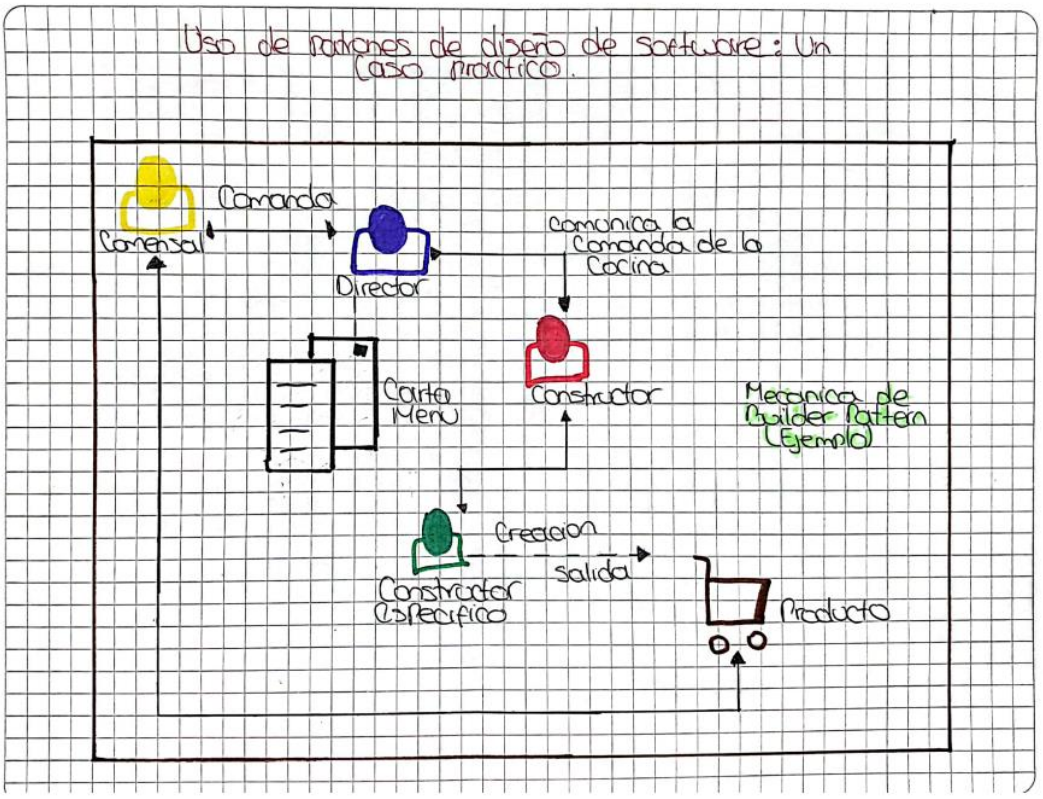
Los patrones de diseño son herramientas fundamentales en la ingeniería de software, ya que brindan soluciones comprobadas a problemas comunes y facilitan el desarrollo eficiente y estructurado de aplicaciones. Entre sus principales beneficios destacan la reutilización del código, la reducción de la complejidad, el desacoplamiento de componentes y la mejora del mantenimiento del software.

En un experimento realizado en la Universidad de Costa Rica, los estudiantes aplicaron patrones de diseño en un proyecto que simulaba un procesador multinúcleo basado en la arquitectura MIPS. Este enfoque permitió ampliar la funcionalidad del simulador, mantener la fidelidad con las arquitecturas reales y desarrollar componentes desacoplados que facilitan el mantenimiento y la extensibilidad. Los resultados demostraron cómo la implementación de patrones puede mejorar significativamente la calidad del software, aunque debe considerarse su impacto en la complejidad del código y las pruebas**.**

Reflexión

Los patrones de diseño son un pilar esencial en la formación de ingenieros de software y científicos de la computación. Incorporar su enseñanza desde etapas tempranas en las carreras universitarias es crucial, ya que promueve el desarrollo de habilidades prácticas que trascienden los métodos tradicionales de enseñanza. Este enfoque no solo capacita a los futuros profesionales para abordar problemas comunes de manera eficiente, sino que también fomenta una mentalidad orientada al diseño y a la adaptabilidad, características clave en un entorno tecnológico en constante cambio.

El experimento descrito evidencia que los patrones de diseño no solo optimizan los proyectos académicos, sino que también preparan a los estudiantes para enfrentar desafíos reales en la industria. Aunque el uso de patrones puede aumentar la complejidad inicial del diseño, los beneficios a largo plazo, como la facilidad de mantenimiento, escalabilidad y flexibilidad, justifican ampliamente su adopción.



Desarrollo de una herramienta para el aprendizaje de patrones de diseño software

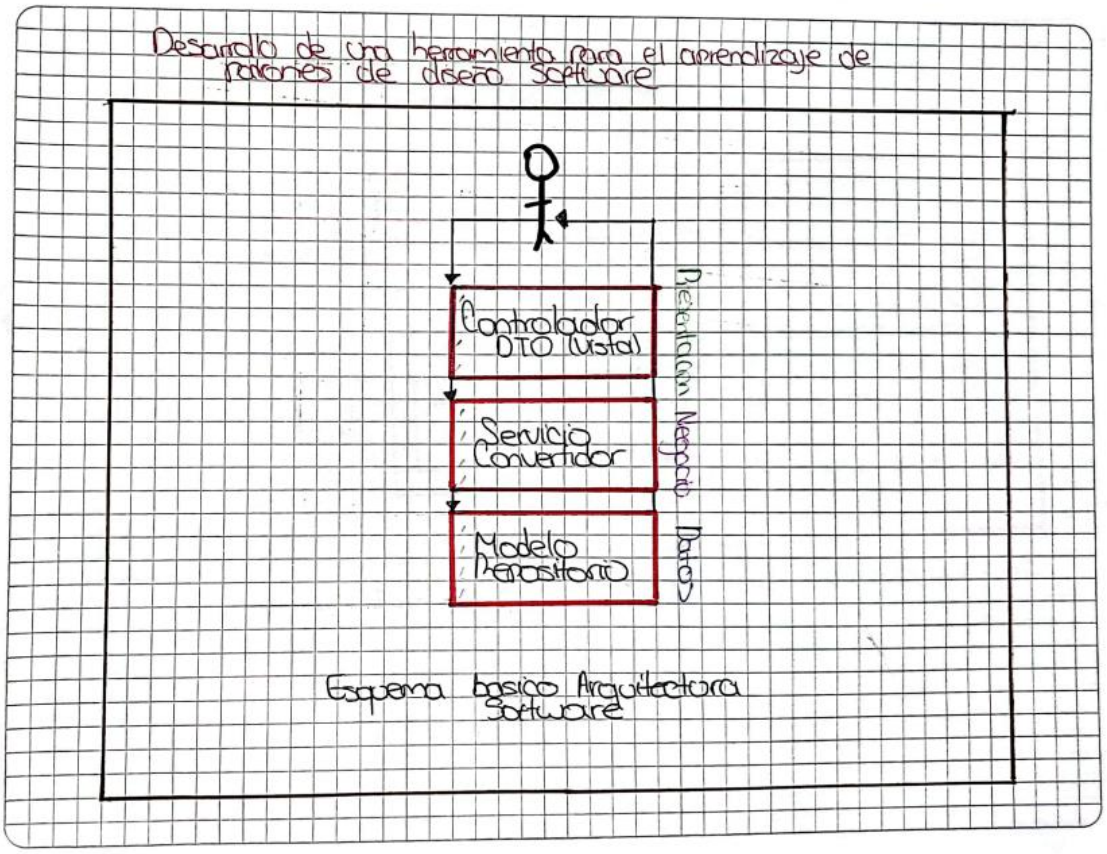
Resumen

Los patrones de diseño ofrecen soluciones probadas y eficaces a problemas recurrentes en programación, facilitando la creación de código limpio y fácil de mantener. Estos patrones no sólo resuelven problemas técnicos, sino que también mejoran la comunicación entre los desarrolladores, ya que el conocimiento de un patrón permite comprender rápidamente su estructura y propósito. El uso de patrones de diseño es crucial para crear aplicaciones bien estructuradas y de alta calidad. Este proyecto busca desarrollar una herramienta que facilite el aprendizaje y uso de patrones de diseño. A pesar de los retos ocasionados por el contexto remoto debido al COVID-19, el trabajo resultó en una aplicación funcional, donde se adquirieron conocimientos tanto en Backend con Spring como en Frontend. El prototipo desarrollado puede ser la base para futuras mejoras y evaluaciones por parte de otros estudiantes.

Reflexión

El desarrollo de una herramienta para aprender patrones de diseño es un paso significativo en la formación de futuros programadores, ya que permite que los principiantes comprendan la importancia de un diseño estructurado desde el inicio. Al adoptar patrones, los desarrolladores logran crear soluciones más robustas y escalables, reduciendo problemas de mantenimiento y mejorando la colaboración en equipo. El aprendizaje de tecnologías como Spring y el desarrollo Frontend, inicialmente desconocidos para el autor, muestra cómo la resolución de problemas a través de la práctica y la investigación puede superar barreras y expandir habilidades. A pesar de la pandemia y el trabajo remoto, el proyecto demostró que con determinación y el uso de recursos adecuados, se pueden lograr resultados satisfactorios. Este prototipo tiene un gran potencial para evolucionar y servir como base para futuras iteraciones que mejoren su funcionalidad y la experiencia del usuario.

Desarrollo de una herramienta para el aprendizaje de patrones de diseño Software



Esquema básico Arquitectura Software

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

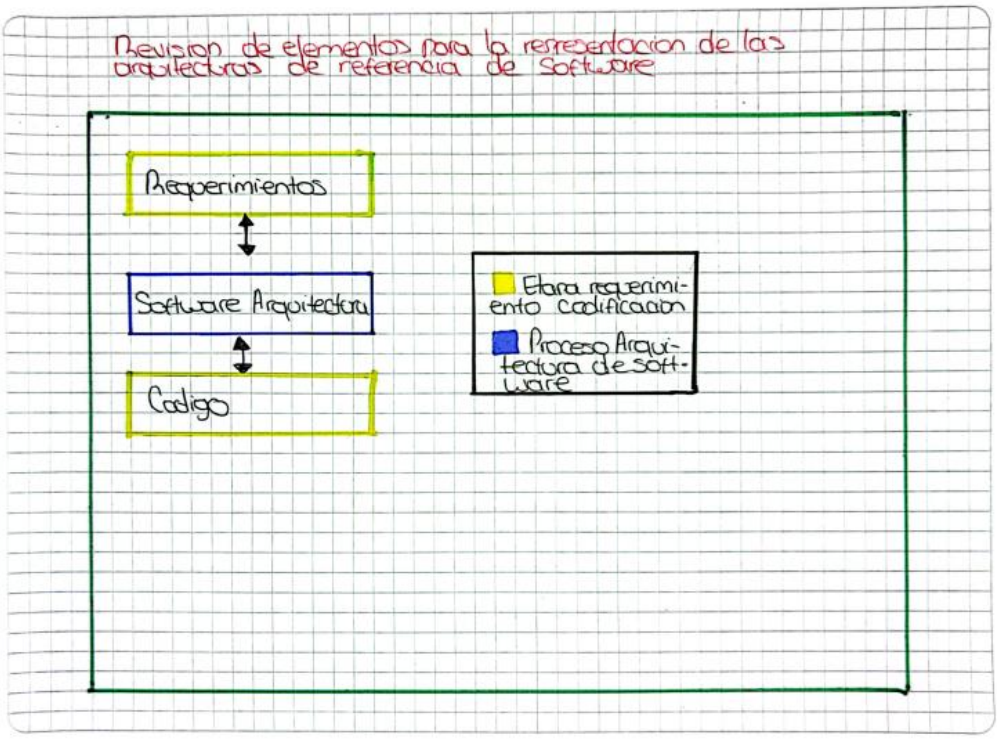
Resumen

La arquitectura de software se enfoca en diseñar y organizar sistemas, incluyendo componentes con relaciones entre ellos. Los patrones guían a los desarrolladores hacia objetivos comunes. Los lenguajes de descripción arquitectónica (ADL) como UniCon, Wright, Darwin, Rapide y C2 ayudan a modelar y analizar arquitecturas antes de implementarlas, optimizando la reutilización, dinámica y evolución. Los componentes y conectores son esenciales en las arquitecturas de software, permitiendo un enfoque modular para el diseño, desarrollo y mantenimiento de sistemas complejos. Las arquitecturas de referencia (ARS) son marcos estándar que mejoran la reutilización y adaptabilidad, integrando conceptos clave para prácticas más eficientes en la industria del software. Estas integran conceptos como configuraciones, restricciones, estilos arquitectónicos y servicios, promoviendo prácticas más eficientes en la industria del software.

Reflexión

La arquitectura de software es esencial para conectar las necesidades del negocio con la implementación técnica, permitiendo una visión completa y modular del sistema. Es importante adaptar las herramientas de modelado y análisis a las necesidades específicas del proyecto y utilizar arquitecturas de referencia para reducir costos y tiempos de desarrollo. Sin embargo, se enfrentan desafíos como la falta de mecanismos para la evolución dinámica de los sistemas y la reutilización efectiva en algunos lenguajes. Comprender y utilizar componentes y conectores es crucial para construir sistemas sólidos, flexibles y sostenibles, fomentando la colaboración interdisciplinaria y abordando los desafíos en un entorno en constante cambio. Este enfoque fomenta la colaboración interdisciplinaria, lo que resulta vital para alcanzar los objetivos del software y abordar sus desafíos en un entorno en constante cambio.

Revision de elementos para la representacion de las arquitecturas de referencia de Software



Atributos de Calidad y Arquitectura de Software

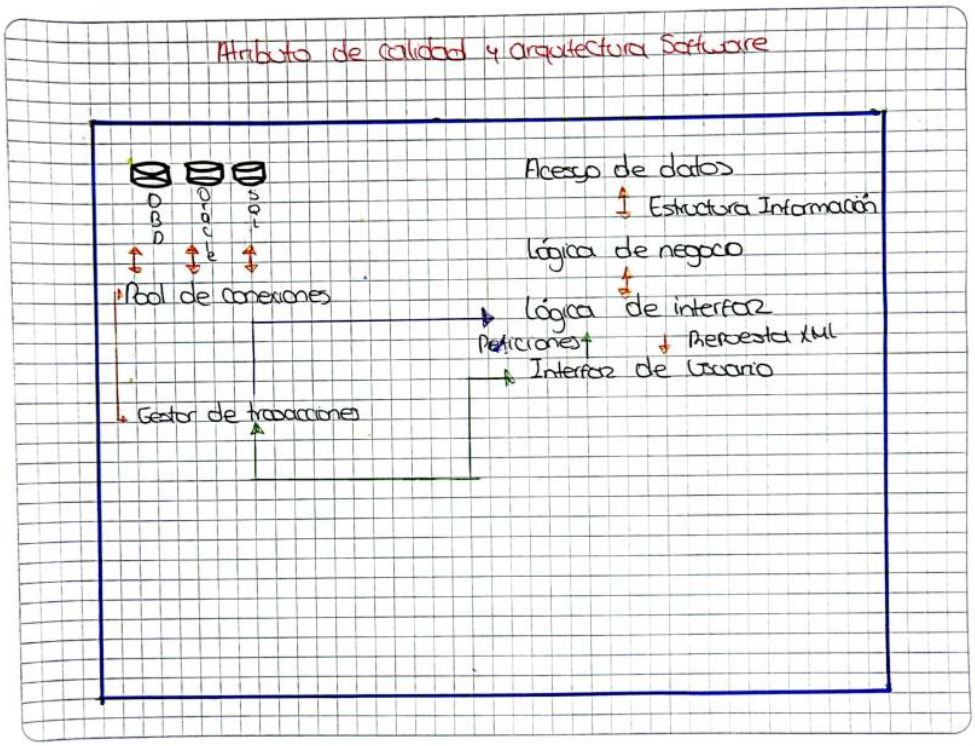
Resumen

En este artículo logramos construir una arquitectura de software efectiva, la cual requiere balancear múltiples atributos de calidad (como seguridad, mantenibilidad, portabilidad y rendimiento) sin comprometer desmedidamente otros. Las decisiones tomadas durante el diseño tienen un impacto significativo en el resultado final. Aplicar una metodología que considere las diferentes perspectivas, documente de manera integral y priorice las necesidades de los stakeholders permite crear arquitecturas que sean tanto funcionales como de calidad, aunque no garantiza estos atributos automáticamente. La práctica de este enfoque se valida mediante casos reales expuestos en el curso. Sin embargo, incluso con un diseño sólido, garantizar los atributos de calidad requiere validación constante y adaptabilidad durante todo el ciclo de vida del sistema.

Reflexión

La arquitectura de software no es solo un conjunto de decisiones técnicas, sino un proceso estratégico que impacta directamente el éxito y la calidad de un sistema. Enfrenta el desafío de equilibrar atributos como rendimiento, seguridad y mantenibilidad, que a menudo entran en conflicto entre sí. Este balance requiere un enfoque sistemático, apoyado en metodologías y herramientas que permitan analizar el sistema desde múltiples perspectivas y documentar de forma clara los compromisos asumidos.

El uso de vistas arquitectónicas y estilos bien definidos no solo facilita la comunicación con los stakeholders, sino que también actúa como una guía para tomar decisiones informadas y mantener la coherencia en el desarrollo. Sin embargo, incluso con un diseño sólido, garantizar los atributos de calidad requiere validación constante y adaptabilidad durante todo el ciclo de vida del sistema.



HERRAMIENTA PARA REUSO DE CÓDIGO JAVASCRIPT ORIENTADO

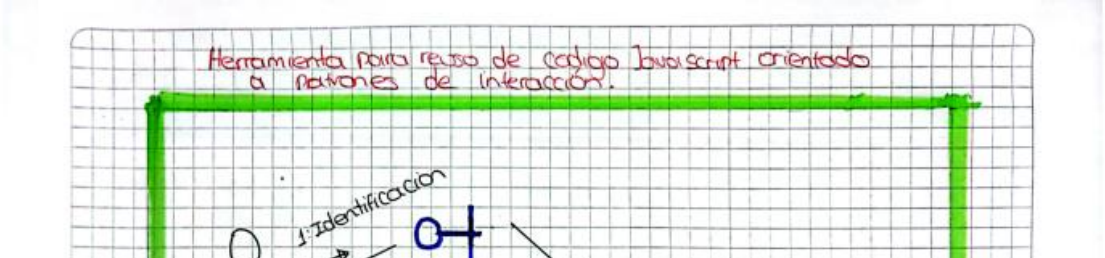
A PATRONES DE INTERACCIÓN

Resumen

El desarrollo de sistemas informáticos es una tarea compleja debido a los diversos problemas que pueden surgir a lo largo de todo el proceso, desde el análisis hasta la implementación. A lo largo del tiempo, han surgido diversas herramientas de diseño rápido de aplicaciones (RAD) que facilitan la creación de interfaces de usuario, como CBuilder, Visual Basic y Delphi. La usabilidad de una interfaz es crucial, ya que determina la efectividad y satisfacción del usuario al interactuar con la aplicación. La reutilización de estos patrones ayuda a acelerar el desarrollo, reducir costos y mejorar la calidad del software. Un ejemplo relevante de la aplicación de esta metodología es ReusMe, una herramienta que permite generar componentes web reutilizables en JavaScript basados en patrones de interacción. Esta aplicación facilita la creación de interfaces gráficas de usuario personalizadas, ofreciendo soluciones probadas y optimizadas para diseñadores y desarrolladores.

Reflexión

El uso de patrones de interacción en el desarrollo de interfaces de usuario y aplicaciones web es una estrategia inteligente para mejorar la eficiencia, reducir el tiempo de desarrollo y garantizar productos de calidad. Al adoptar patrones reutilizables, los diseñadores no solo evitan "reinventar la rueda", sino que también aprovechan las mejores prácticas y soluciones que han sido probadas a lo largo del tiempo. Además, herramientas como ReusMe demuestran el valor de la reutilización de código en la creación de interfaces gráficas, ya que facilitan la personalización y adaptación de componentes a las necesidades del usuario, sin sacrificar la usabilidad y funcionalidad. En resumen, el empleo de patrones y la reutilización de código son pasos fundamentales para optimizar el proceso de desarrollo y ofrecer productos más robustos y accesibles**.**



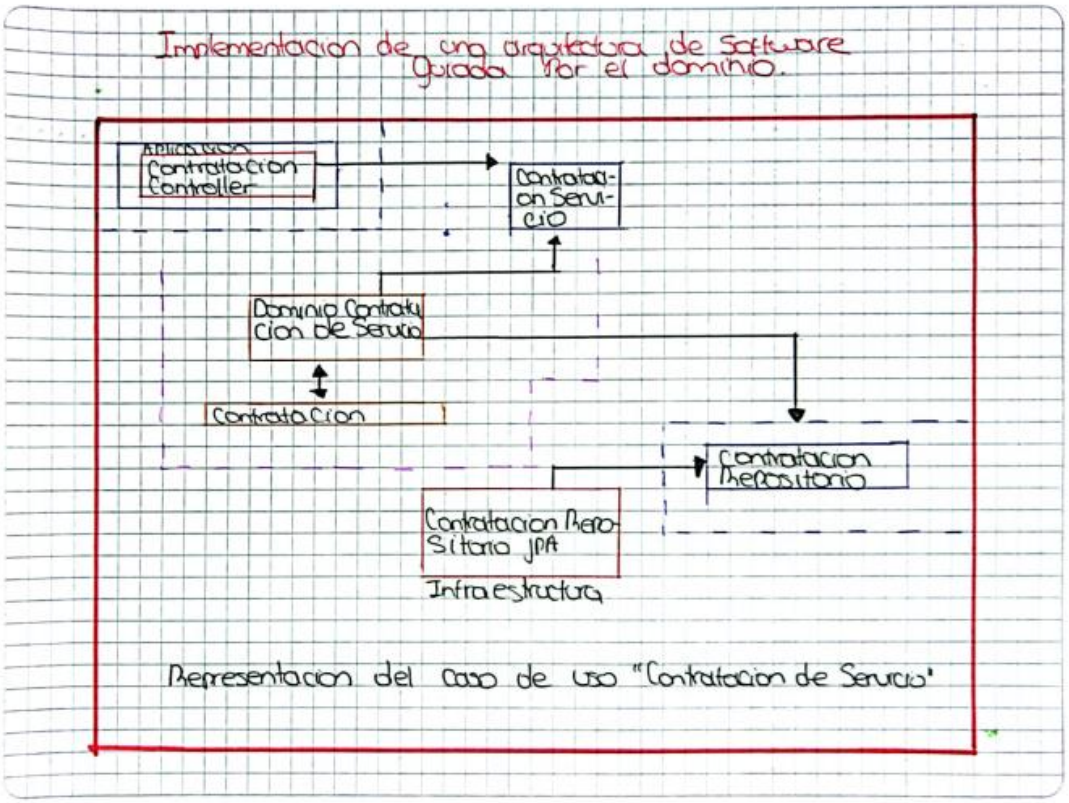
Implementación de una Arquitectura de Software
guiada por el Dominio

Resumen

Presenta un enfoque de implementación de arquitecturas de software guiadas por el dominio, enfatizando el diseño dirigido por el dominio (DDD). Se propone transformar una arquitectura de software típica en una arquitectura hexagonal, centrada en el dominio del negocio. Esto permite separar la complejidad del negocio de la lógica técnica, promoviendo una mejor comunicación entre expertos en dominio y desarrolladores. Se discuten los conceptos de contextos delimitados y lenguaje ubicuo, fundamentales para el DDD. Además, se valida el enfoque a través de un caso de estudio sobre una plataforma de empleo, donde se ejemplifica cómo aplicar estos principios en un entorno real. La investigación concluye que esta transformación mejora la mantenibilidad y escalabilidad del software.

Reflexión

El enfoque presentado en el artículo es muy relevante en el contexto actual del desarrollo de software, donde la complejidad y la rapidez de cambio son constantes. La adopción de una arquitectura guiada por el dominio no solo facilita una mejor alineación con los objetivos de negocio, sino que también promueve una colaboración más efectiva entre diferentes partes interesadas. La transformación hacia una arquitectura hexagonal, que favorece la independencia de tecnologías, es una estrategia inteligente para asegurar la adaptabilidad a futuros cambios y mejorar la calidad del software. Sin embargo, la implementación de estos conceptos puede ser desafiante y requiere un compromiso significativo por parte de todo el equipo de desarrollo.



Resumen

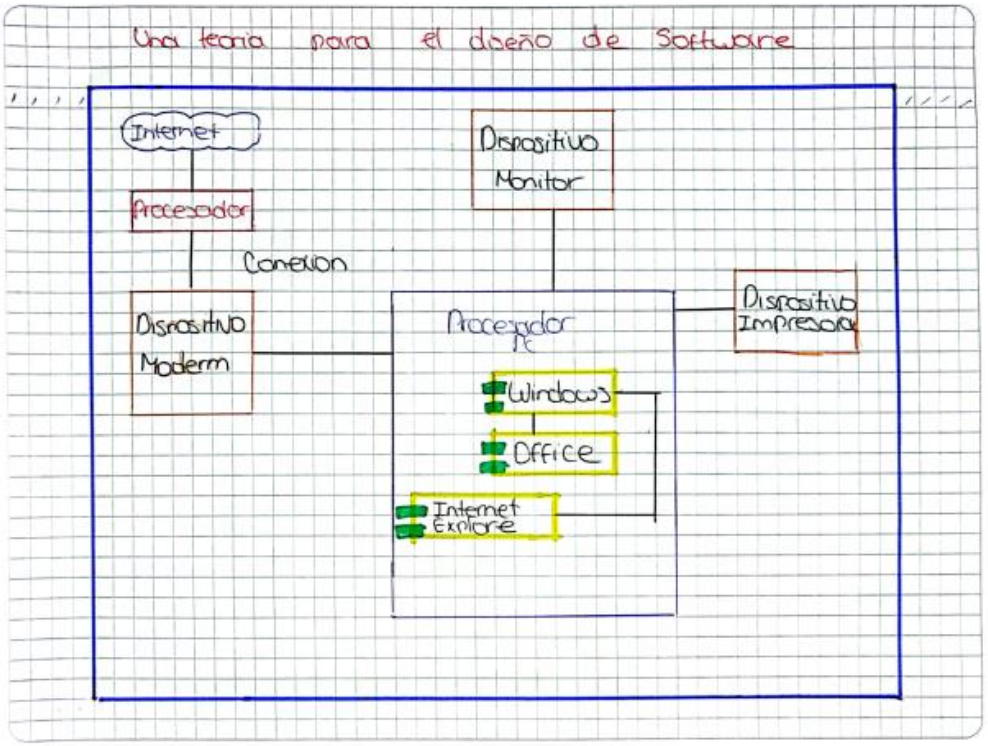
El diseño de software es esencial porque, al dividir un sistema en elementos que interactúan entre sí, se facilita su desarrollo y mantenimiento. Este enfoque permite asignar funciones a cada componente, establecer sus relaciones y describir su estructura, lo que reduce el costo y la complejidad a lo largo del ciclo de vida del software.

El proceso de producción de software se divide en dos etapas principales: desarrollo (33% del esfuerzo total) y mantenimiento (67%). El mantenimiento, que incluye cambios, correcciones y mejoras, suele ser la parte más costosa, especialmente por la necesidad de re-testear el sistema cada vez que se introduce un cambio.

El diseño de software se vuelve crucial para reducir los costos de mantenimiento, ya que anticipa y facilita la incorporación de cambios sin comprometer la integridad del sistema. Un buen diseño permite que los cambios se realicen con menor costo, ya que organiza y descompone el código de manera eficiente.

Reflexión

La reflexión sobre el diseño de software nos muestra la importancia de planificar y estructurar adecuadamente un sistema desde el principio. Aunque pueda parecer tentador lanzarse directamente a la programación, sin un diseño previo, los costos y problemas que surgen a lo largo del ciclo de vida del software pueden ser mucho mayores. El principio de "Diseño para el Cambio" nos recuerda que el software no es algo estático; debe ser flexible y evolucionar con el tiempo. En definitiva, el diseño adecuado no es solo una cuestión técnica, sino una estrategia económica que puede hacer la diferencia entre un sistema costoso de mantener y uno que se adapte con facilidad a las necesidades futuras.



Desarrollo de una arquitectura de software para el robot móvil Lázaro

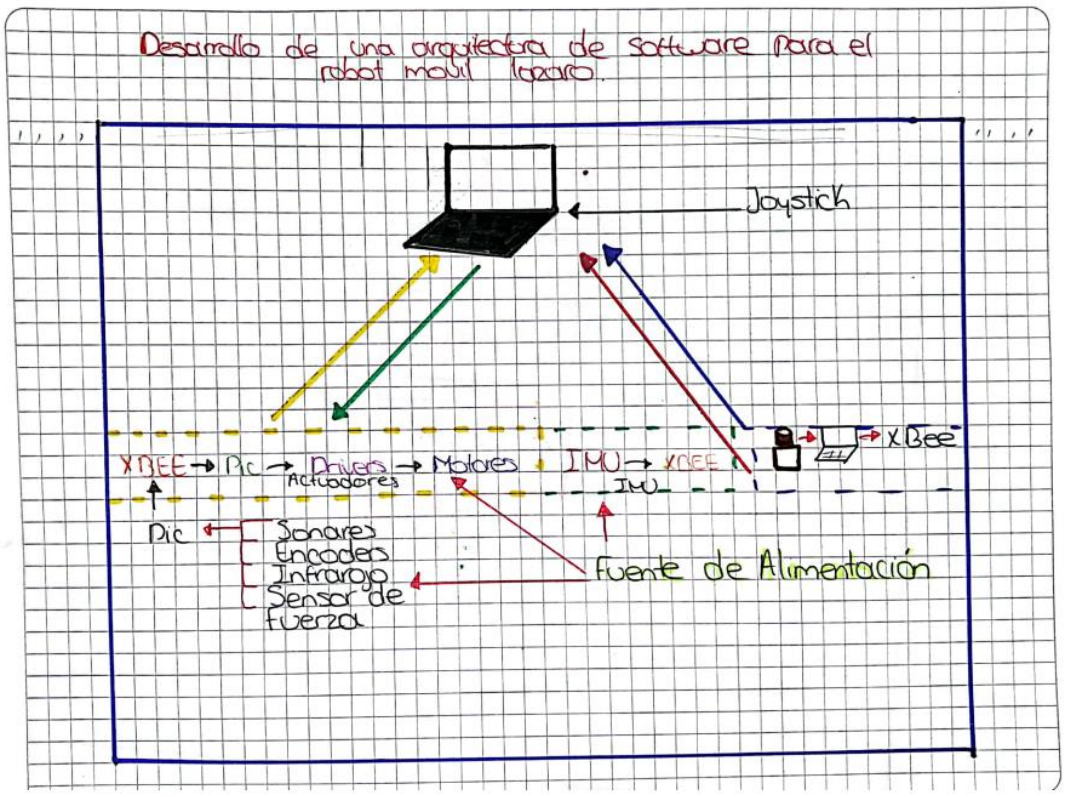
Resumen

presenta una innovadora estructura de software diseñada para optimizar el control y la operación del robot Lázaro. La arquitectura se compone de tres niveles que permiten la gestión eficiente de los actuadores y el monitoreo de los sensores, utilizando librerías implementadas en lenguaje C#. Se incorporan diversos sensores, como sonares y un telémetro láser, para la detección de obstáculos y la medición de fuerzas. La comunicación entre el robot y un computador remoto se realiza a través de módulos XBee®, permitiendo un control tanto autónomo como teleoperado. Además, se discuten las ventajas de arquitecturas deliberativas y reactivas, así como la implementación de herramientas de inteligencia artificial para mejorar la navegación en entornos no estructurados. La evaluación cualitativa de la arquitectura muestra resultados positivos en términos de flexibilidad, reactividad y facilidad de uso, aunque se identifican áreas de mejora, como el aprendizaje de tareas. En conclusión, la arquitectura propuesta es escalable y amigable para los usuarios, lo que la hace adecuada para las necesidades operativas del robot.

Reflexión

La propuesta de una arquitectura de software para el robot Lázaro es un avance significativo en el campo de la robótica móvil. La integración de múltiples sensores y la capacidad de control remoto ofrecen un enfoque versátil y adaptable a diferentes entornos. Además, la combinación de arquitecturas deliberativas y reactivas, junto con el uso de inteligencia artificial, demuestra un entendimiento profundo de los desafíos que enfrentan los robots en situaciones dinámicas. Sin embargo, sería interesante ver cómo se pueden abordar las limitaciones en el aprendizaje de tareas, lo que podría abrir nuevas posibilidades para la autonomía del robot. En general, este trabajo no solo contribuye al desarrollo del robot Lázaro, sino que también sienta las bases para futuras investigaciones en arquitecturas de software para robots móviles.

Desarrollo de una arquitectura de software para el robot movil Lazaro.



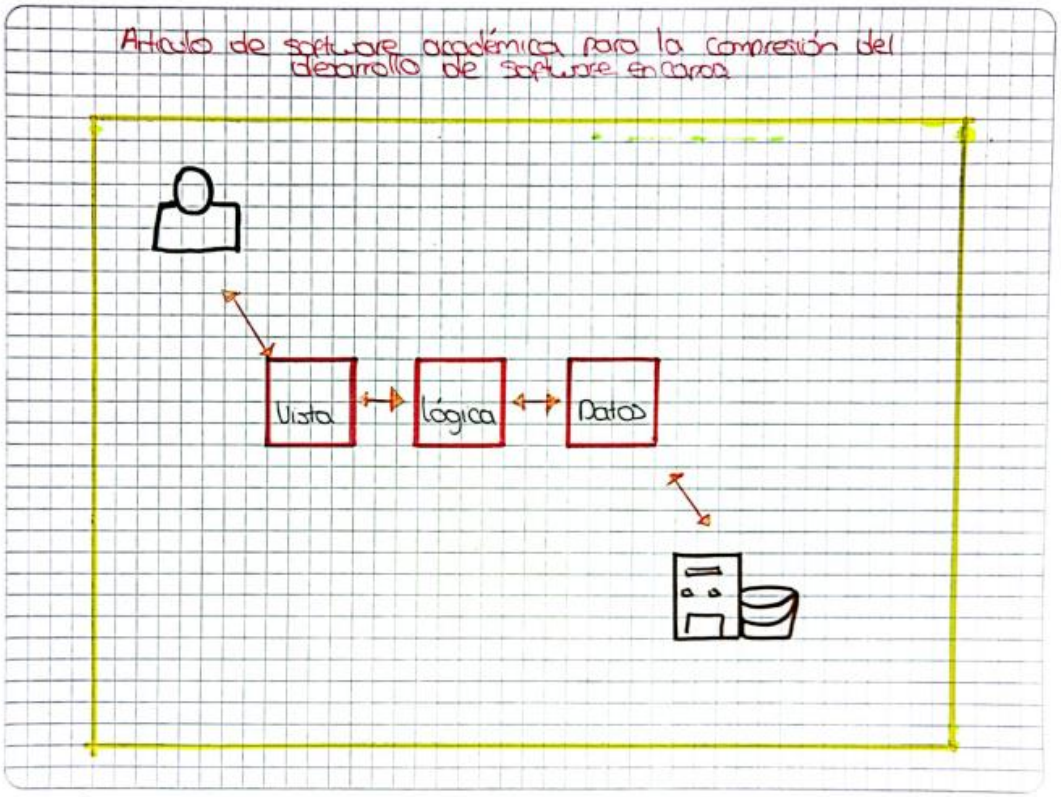
Articulo de software académica para la comprensión del desarrollo de software en capas

Resumen

Este artículo expone un modelo académico de arquitectura de software basado en un enfoque de desarrollo en capas, destacando su relevancia para prevenir problemas a corto y largo plazo. Propone una estructura de tres capas (Vista, Lógica y Datos) y una más detallada de siete capas (Interfaz Gráfica de Usuario, Controlador, Interfaces, Estructura, Lógica, Mapeo Objeto-Relacional y Acceso a Datos), para optimizar el desarrollo de sistemas. A través del patrón de diseño Modelo-Vista-Controlador (MVC), se gestionan las interacciones entre la interfaz gráfica y las capas del sistema, promoviendo una separación clara de responsabilidades, lo que facilita la reutilización y el mantenimiento del código. Se describen clases como Cliente y Teléfono Datos, que implementan la interfaz IABMC para operaciones sobre la base de datos, y la clase Comando, encargada de las interacciones con la base de datos. El artículo concluye que la adopción de arquitecturas de software es clave para desarrollar sistemas flexibles y sostenibles, sugiriendo mejoras como la automatización de mapeadores y el control de transacciones.

Reflexión

El texto resalta la importancia de las arquitecturas de software en capas, utilizando el patrón Modelo-Vista-Controlador (MVC) para organizar y mejorar la reutilización y el mantenimiento del código. Presenta ejemplos prácticos, como las clases Cliente y Teléfono Datos, para ilustrar su aplicación. Aunque es un enfoque sólido, se sugiere profundizar en la automatización de mapeadores y el control de transacciones para mejorar la flexibilidad del sistema. En general, el texto subraya cómo una arquitectura bien estructurada favorece la creación de sistemas más sostenibles y fáciles de mantener.



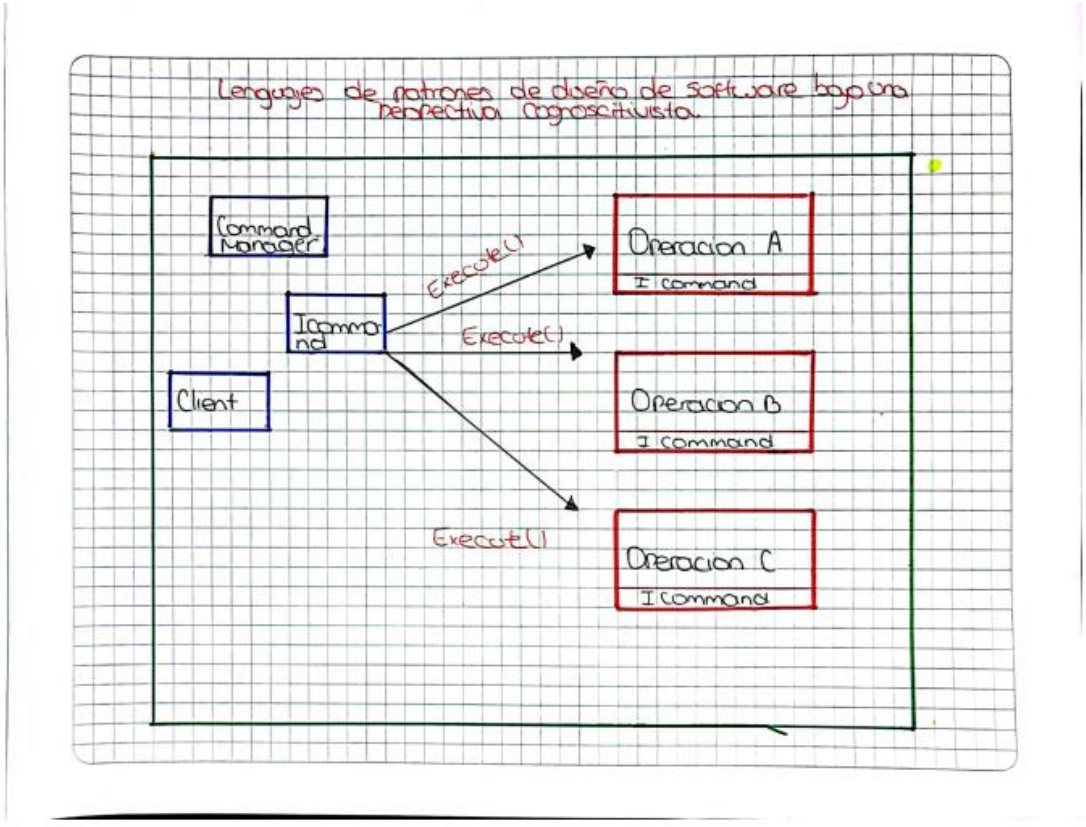
Lenguajes de patrones de diseño de software bajo una perspectiva cognoscitivista.

Resumen

Este artículo nos ayuda analizar la aplicación del concepto de "lenguaje de patrones" en el diseño de software, basándose en las ideas de Alexander (1979) y su relación con la estructura de sistemas de patrones. Aunque el término "lenguaje de patrones" se ha utilizado ampliamente, aún persisten dudas teóricas y prácticas sobre su significado y aplicación. El trabajo propone un marco teórico cognoscitivo para interpretar y mejorar el uso de estos lenguajes. Se establece una distinción clara entre los conceptos de "lenguaje de patrones", "estructura de lenguaje de patrones", "catálogos de patrones" y "sistemas de patrones". Además, se sugiere que el diseño de un lenguaje de patrones debe enfocarse en su organización y evolución para mejorar su eficacia en el campo del desarrollo de software.

Reflexión

La reflexión que nos deja este artículo resalta la complejidad y la importancia de los lenguajes de patrones en el diseño de software, señalando que aunque el concepto ha sido ampliamente utilizado, su comprensión aún carece de una base teórica consolidada. Esta falta de claridad teórica podría dificultar su implementación efectiva. La propuesta de un marco cognoscitivo para organizar y mejorar estos lenguajes refleja una necesidad crítica de estructurar el conocimiento sobre patrones de manera más efectiva. Además, el enfoque en la evolución y organización del conocimiento sobre patrones apunta a una mejora continua en su aplicabilidad, lo que puede ser clave para avanzar en el campo del desarrollo de software. Este trabajo también sugiere nuevas direcciones para futuras investigaciones, lo que subraya la importancia de seguir innovando y clarificando conceptos fundamentales en la ingeniería de software.



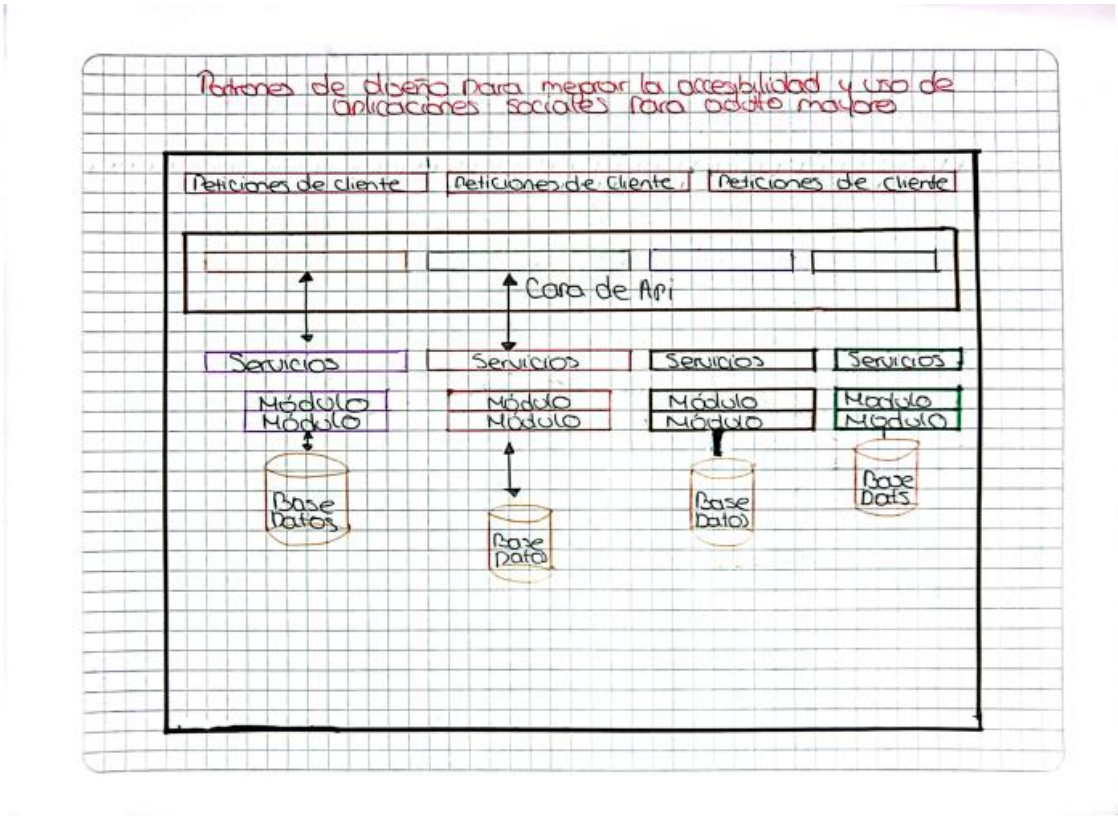
Patrones de diseño para mejorar la accesibilidad y uso de aplicaciones sociales para adultos mayores.

Resumen

El propósito de este artículo es proponer un conjunto incompleto de 36 modelos para el diseño de interacción en aplicaciones sociales dirigidas a personas mayores. La propuesta se basa en esfuerzos previos, a menudo expresados como normas y directrices de diseño, y busca identificar posibles problemas de usabilidad en dichas interfaces. Además, se centra en fortalecer estos métodos mediante la inclusión de descripciones de anomalías y soluciones alternativas dentro de la estructura del modelo, que los diseñadores y desarrolladores pueden utilizar. Para lograrlo, se emplean técnicas de "evaluación heurística" (usualmente utilizadas en la interacción persona-computadora) para obtener la percepción del usuario sobre un diseño en particular. El trabajo aborda el tema desde dos perspectivas: la del técnico y la del grupo social de personas mayores. Los resultados muestran que el modelo propuesto facilita la creación de interfaces bien diseñadas, que brindan una mejor experiencia de usuario y tienen un impacto positivo en la calidad de vida de las personas mayores.

Reflexión

La propuesta presentada en este artículo resalta la importancia de crear interfaces tecnológicas accesibles y usables para las personas mayores, un grupo que a menudo enfrenta barreras en el uso de aplicaciones sociales debido a limitaciones cognitivas y físicas. Al integrar un conjunto de modelos de interacción y técnicas de evaluación heurística, se ofrece un enfoque centrado tanto en la perspectiva técnica como en las necesidades sociales de este colectivo, con el fin de promover la adopción de la tecnología y mejorar la calidad de vida de los usuarios mayores. Este enfoque no solo mejora la experiencia de usuario, sino que también fomenta el bienestar emocional y social al facilitar la conectividad y la participación en la sociedad digital. En definitiva, el artículo enfatiza la necesidad de continuar desarrollando diseños inclusivos que permitan a las personas mayores aprovechar las ventajas de la tecnología, contribuyendo a su empoderamiento y reduciendo la exclusión digital.



Arquitectura de software de una aplicación móvil para desarrollar un sistema de identificación por radiofrecuencia

Resumen

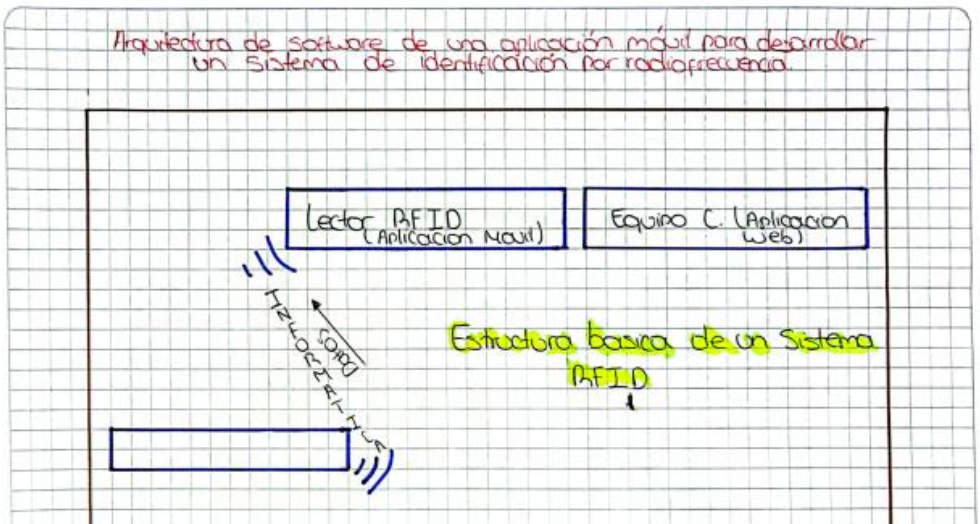
Los sistemas de identificación por radiofrecuencia (RFID) se utilizan para identificar objetos automáticamente, y en este caso, se está desarrollando un sistema RFID para

mejorar el control de inventarios en el Instituto Tecnológico de Orizaba. Actualmente, el sistema de inventarios del instituto presenta problemas como la falta de concordancia entre los activos registrados y los realmente existentes, así como un tiempo de actualización lento. El sistema RFID propuesto busca resolver estos problemas al disminuir el tiempo de registro, actualización y localización de los activos. La arquitectura de software del sistema incluye una base de datos XML para integrar la aplicación web y la aplicación RFID, y un módulo RFID que envía señales de radiofrecuencia para detectar objetos etiquetados. La implementación de este sistema incrementa la seguridad y el control de los bienes, y permite mantener el inventario actualizado en tiempo real, evitando errores humanos y robos.

Reflexión

El uso de tecnología RFID para el control de inventarios en instituciones como el Instituto Tecnológico de Orizaba representa un avance significativo en la eficiencia y precisión en la gestión de activos. Este sistema no solo mejora la actualización y consulta de los inventarios, sino que también facilita la localización de los bienes de manera más rápida y precisa, reduciendo el margen de error humano. La integración de aplicaciones web y móviles con tecnologías como RFID muestra cómo la automatización y la digitalización pueden transformar procesos tradicionales, brindando mayor seguridad y eficiencia. Además, la implementación de un sistema de este tipo no solo mejora la gestión de los activos materiales, sino que también contribuye a una mejor organización interna, permitiendo a las instituciones optimizar sus recursos. Este tipo de innovación es crucial para avanzar hacia un modelo de gestión más moderno y eficiente en el ámbito educativo y organizacional. Sin embargo, también es importante considerar los costos iniciales de implementación y la capacitación necesaria para el personal, lo que podría representar un desafío en términos de inversión y adaptación.

Arquitectura de software de una aplicación móvil para desarrollar un Sistema de Identificación por radiofrecuencia.



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

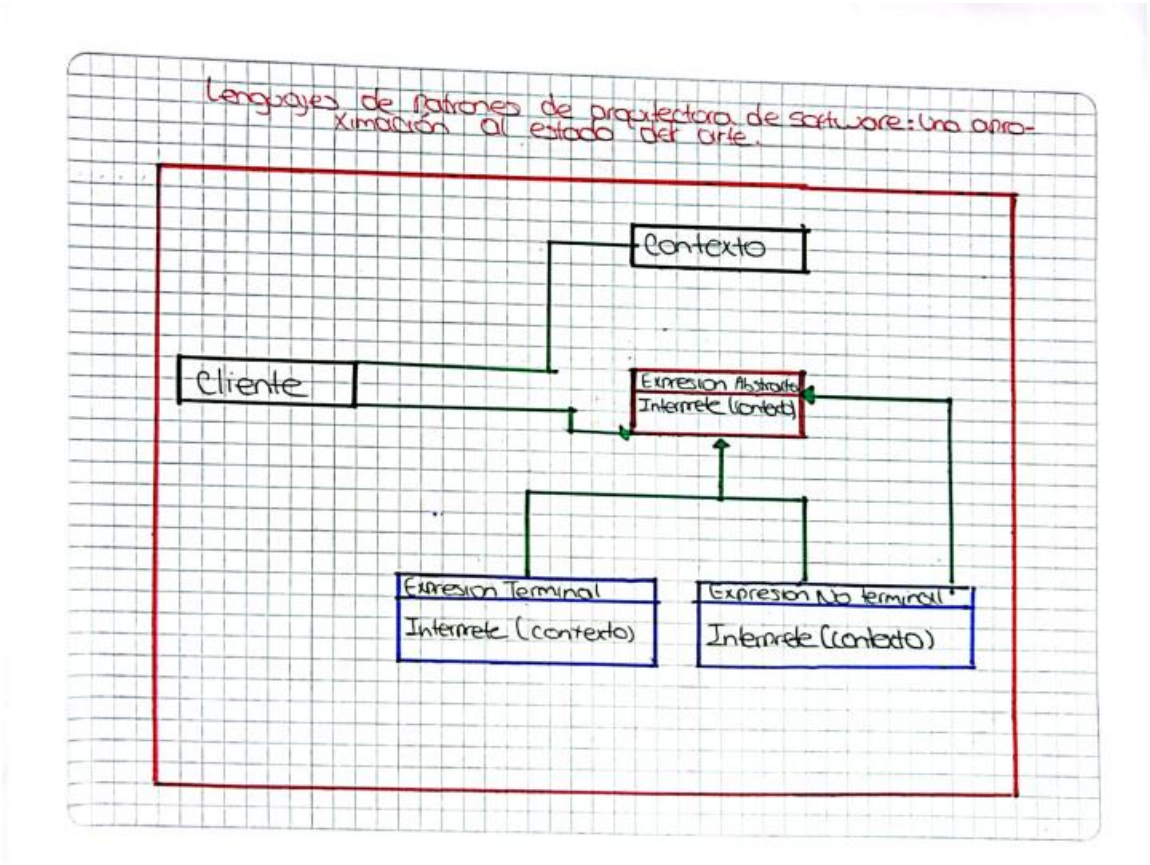
Resumen

Examina el estado actual de los lenguajes de patrones en la arquitectura de software. Se analizan sus orígenes, avances y aplicaciones en la construcción de arquitecturas en

diversos dominios, destacando su importancia para diseñadores y desarrolladores. Se revisa la evolución de la ingeniería de software desde los años 60, enfatizando la necesidad de un enfoque arquitectónico para mejorar la calidad y mantenibilidad del software. El artículo menciona a Christopher Alexander y Frank Buschmann, quienes contribuyeron al desarrollo de patrones en la arquitectura civil y de software, respectivamente. Se presentan diferentes formas de categorizar patrones y ejemplos de su aplicación en áreas como la gestión de identidades y el desarrollo de aplicaciones e-business. Se concluye que los lenguajes de patrones son herramientas valiosas para resolver problemas arquitectónicos de manera eficiente y se sugiere investigar más sobre metodologías en su creación.

Reflexión

El artículo ofrece una visión clara sobre los lenguajes de patrones en la arquitectura de software, destacando su evolución y relevancia en el desarrollo de sistemas mantenibles y de alta calidad. Conecta sus orígenes, desde los aportes de Christopher Alexander en la arquitectura civil hasta su adaptación por Frank Buschmann al ámbito del software, lo que proporciona una base histórica sólida. Además, presenta ejemplos prácticos, como su aplicación en e-business y gestión de identidades, lo que refleja la versatilidad de estos patrones. La invitación a investigar metodologías para su creación fomenta el avance en su uso sistemático, resaltando la importancia de estos lenguajes como soluciones eficientes para problemas arquitectónicos complejos



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

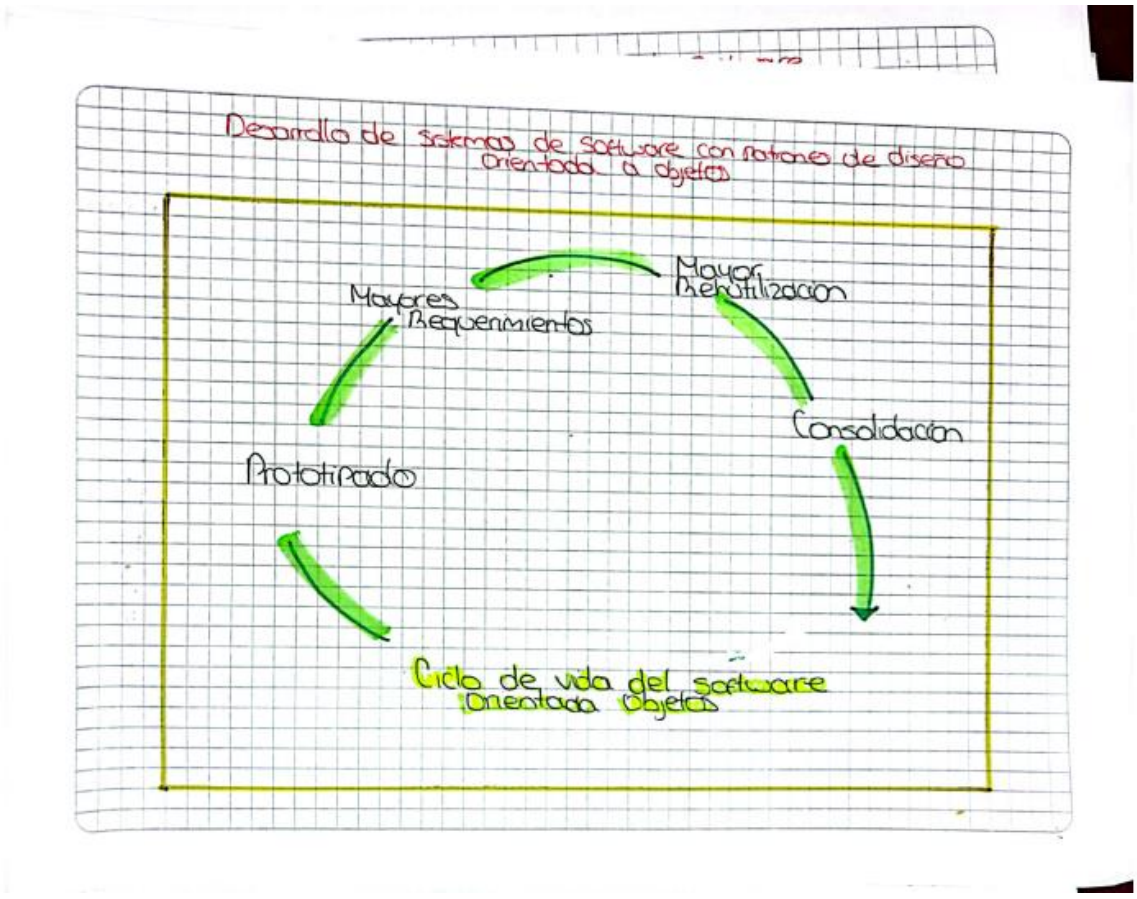
Resumen

Aborda la implementación de patrones de diseño orientados a objetos en la creación de un sistema de software denominado "Intranet Industrial", desarrollado en la Facultad de Ingeniería Industrial de la Universidad Nacional Mayor de San Marcos. Se analiza el impacto económico de la adopción de estos patrones en comparación con un enfoque que

no los considera. En la primera sección, se introducen conceptos teóricos y se clasifican los patrones más reconocidos en el ámbito industrial. A continuación, se detallan los patrones aplicados en el desarrollo del sistema y se presentan de manera formal. La sección final ofrece una descripción del sistema, sus módulos y herramientas, así como una estimación de costos utilizando el patrón "Informador" y el método COCOMO. Se concluye que la implementación de patrones contribuye a mejorar la eficiencia y a disminuir los costos en el desarrollo de software.

Reflexión

El enfoque de utilizar patrones de diseño en el desarrollo de software es una estrategia altamente efectiva que no solo optimiza el proceso de creación, sino que también promueve la reutilización y la escalabilidad. Este trabajo es un excelente ejemplo de cómo integrar teoría y práctica, mostrando que la implementación de patrones no solo es beneficiosa desde el punto de vista técnico, sino que también tiene repercusiones positivas en la gestión de costos. La investigación resalta la importancia de adoptar estándares en el desarrollo de sistemas complejos, ofreciendo un modelo que puede servir de referencia para futuros proyectos en el ámbito académico e industrial.



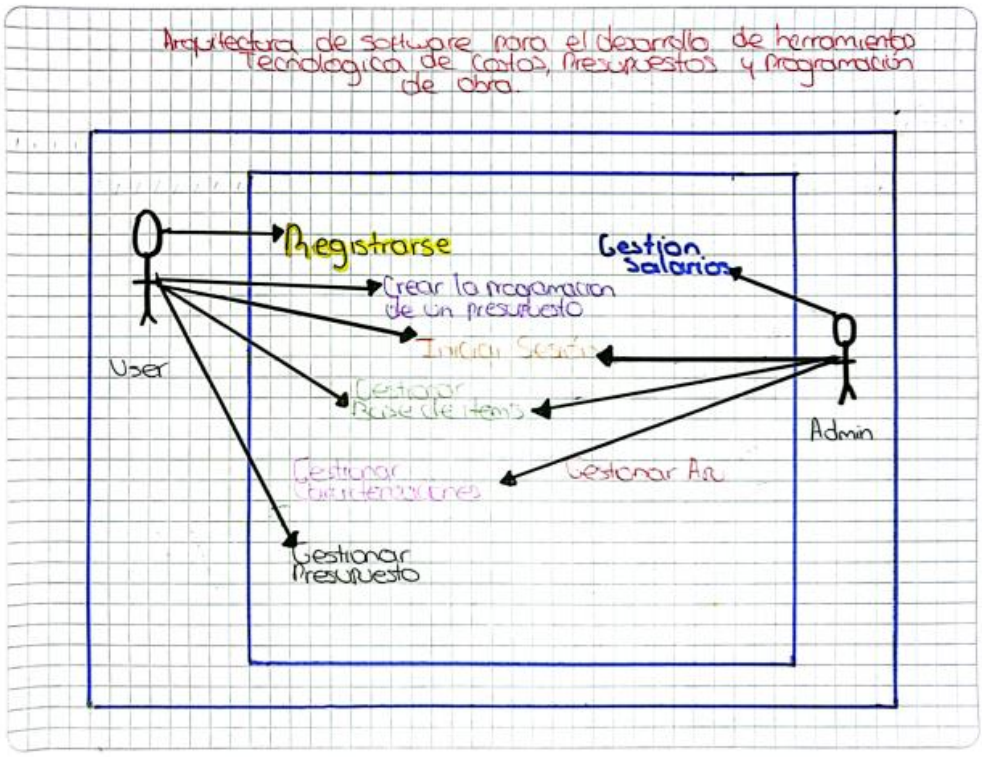
Arquitectura de software para el desarrollo de herramienta Tecnológica de Costos, Presupuestos y Programación de obra

Resumen

El artículo "Arquitectura de software para el desarrollo de herramienta Tecnológica de Costos, Presupuestos y Programación de obra" de Cárdenas-Gutiérrez, Barrientos-Monsalve y Molina-Salazar, se centra en el diseño de un software académico para la gestión de costos y presupuestos en Ingeniería Civil. Se organiza la información relacionada con materiales, mano de obra, maquinaria y transporte en grupos de procesos, buscando aumentar la productividad y facilitar la elaboración de presupuestos. La metodología incluye la creación de una Estructura de División del Trabajo (EDT) y el uso del método de la ruta crítica para la programación de obras. Se presentan casos de uso del software, que incluyen la gestión de presupuestos y salarios. La propuesta tiene como objetivo modernizar la enseñanza y mejorar la calidad educativa, ofreciendo una herramienta accesible y gratuita para los estudiantes. Los autores no tienen conflictos de intereses y el artículo concluye que esta herramienta puede preparar mejor a los estudiantes para el mercado laboral.

Reflexión

Me parece interesante porque propone un software práctico y accesible, pensado para facilitar la gestión de costos, presupuestos y programación. Me gusta que utilice metodologías como la EDT y el método de la ruta crítica, ya que ayudan a conectar la teoría con la práctica. Además, que sea gratuito lo hace aún más valioso. Solo me pregunto cómo se asegura su actualización y si ya ha sido probado con usuarios.



Una arquitectura basada en software libre para archivos web

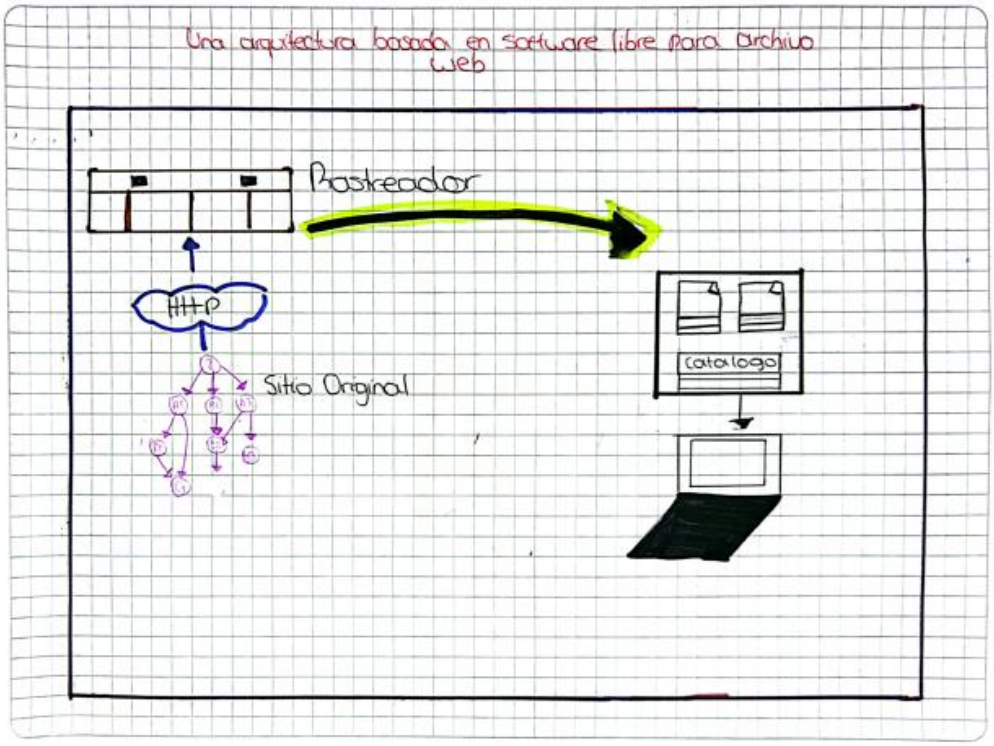
Resumen

La información en la web es extensa pero puede ser eliminada. Un documento web está compuesto por archivos HTML y otros elementos. Desde los años 90 se han creado

sistemas para preservar sitios web, conocidos como archivos web. En Latinoamérica y Venezuela hay falta de iniciativas de preservación web. Se busca diseñar una arquitectura que facilite la preservación web utilizando software libre. Se realizó un estudio de métodos y herramientas de preservación web a nivel mundial. Se seleccionaron los más adecuados para la arquitectura propuesta, priorizando el uso de software libre. Se está desarrollando un prototipo para probar la integración y rendimiento de estos componentes. La preservación de la información web es crucial para garantizar el acceso continuo a recursos digitales y proteger el patrimonio cultural. El objetivo final es proporcionar un marco de referencia para implementar sistemas de preservación web efectivos y sostenibles en Venezuela y Latinoamérica.

Reflexión

La información en la web es extensa pero puede ser eliminada. Un documento web está compuesto por archivos HTML y otros elementos. Desde los años 90 se han creado sistemas para preservar sitios web, conocidos como archivos web. En Latinoamérica y Venezuela hay falta de iniciativas de preservación web. Se busca diseñar una arquitectura que facilite la preservación web utilizando software libre. Se realizó un estudio de métodos y herramientas de preservación web a nivel mundial. Se seleccionaron los más adecuados para la arquitectura propuesta, priorizando el uso de software libre. Se está desarrollando un prototipo para probar la integración y rendimiento de estos componentes. La preservación de la información web es crucial para garantizar el acceso continuo a recursos digitales y proteger el patrimonio cultural. El objetivo final es proporcionar un marco de referencia para implementar sistemas de preservación web efectivos y sostenibles en Venezuela y Latinoamérica.



Patrones de adaptación para arquitecturas de software basadas en tecnologías del acuerdo

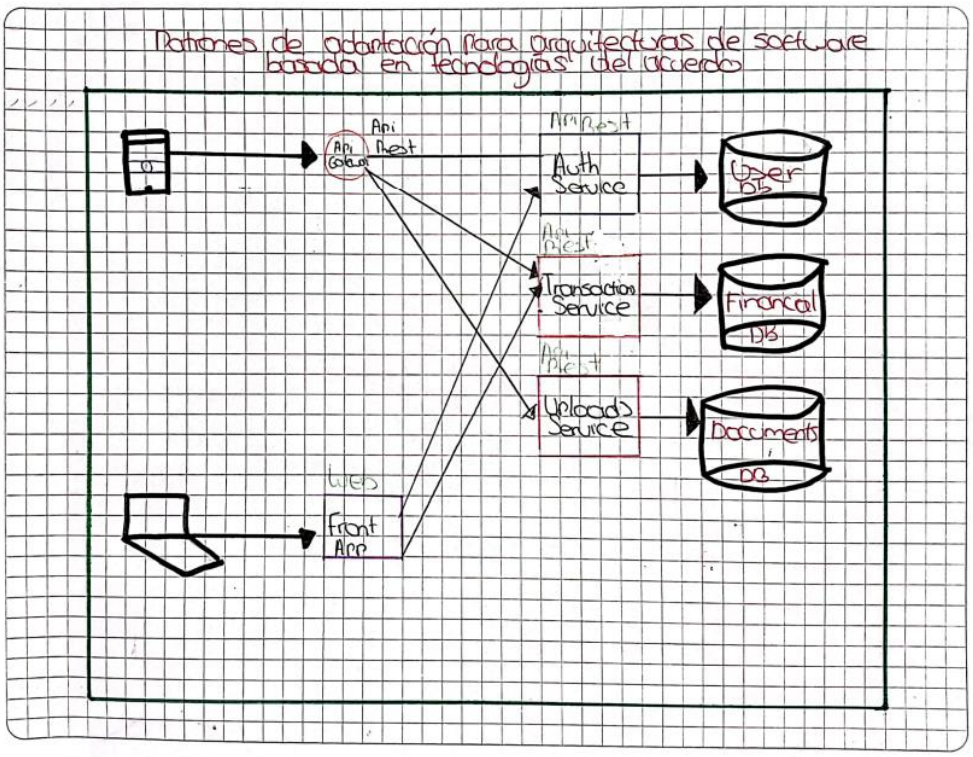
Resumen

Los sistemas actuales son cada vez más complejos y requieren un enfoque adaptativo para gestionar su evolución. La auto-adaptación, que no solo abarca el comportamiento funcional, sino también las propiedades no-funcionales, es esencial en estos sistemas. Los Sistemas Multi-Agente (SMA) han surgido como una solución para resolver problemas complejos, pero aún presentan limitaciones en cuanto a su auto-adaptación. La propuesta presentada se basa en un ecosistema de servicios con Tecnologías del Acuerdo (AT), donde los agentes se organizan y coordinan dinámicamente, formando organizaciones adaptativas. Estas organizaciones evolucionan mediante reglas de adaptación y patrones predefinidos, alcanzando acuerdos estables. La plataforma THOMAS es la base para este enfoque, que busca desarrollar una arquitectura adaptativa a través de la evolución de los agentes y sus interacciones. El futuro del proyecto incluye la implementación de nuevas variantes para mejorar la adaptabilidad de los sistemas.

Reflexión

La capacidad de auto-adaptación es fundamental para los sistemas modernos, ya que permiten una mayor flexibilidad y eficiencia frente a cambios en su entorno o en los requerimientos. El enfoque basado en agentes y acuerdos emergentes tiene un gran potencial, pues permite que los sistemas evolucionen de manera autónoma sin intervención humana constante. Sin embargo, aún queda un largo camino por recorrer para alcanzar una autoadaptación verdadera y completamente dinámica. El desarrollo de patrones de adaptación y su integración en plataformas como THOMAS es un paso crucial en la creación de arquitecturas que puedan auto-organizarse y adaptarse en tiempo real. Este enfoque no solo mejora la eficiencia, sino que también proporciona una base sólida para afrontar la complejidad creciente de los sistemas, permitiendo que los mismos se ajusten a nuevas demandas sin perder estabilidad ni funcionalidad. La experimentación

y refinamiento continuo en plataformas como ATMAS serán clave para lograr la meta de sistemas completamente auto-adaptativos.



¿CÓMO EMPLEAR EL SOFTWARE LIBRE EN LA ARQUITECTURA Y EL DISEÑO?

Resumen

Este trabajo expone el uso del software libre en disciplinas fuera de la informática, específicamente en Arquitectura y Diseño. El enfoque destaca cómo estas herramientas pueden mejorar la calidad de los trabajos realizados con recursos limitados. Además, se presenta una guía básica para quienes se inician en el uso de software libre, orientada a aquellos sin conocimientos previos, sugiriendo programas que pueden sustituir a software propietario. El autor muestra ejemplos prácticos que demuestran las capacidades y ventajas de estas herramientas en áreas como el diseño arquitectónico e industrial. El software libre fomenta el intercambio de ideas y la creación de modelos de trabajo que promueven la libertad y creatividad en el proceso de diseño.

Reflexión

El software libre representa una revolución no solo en el ámbito tecnológico, sino también en campos como la Arquitectura y el Diseño, donde la creatividad y la accesibilidad son cruciales. Al ser accesible y modificable, el software libre permite que más personas participen en el proceso de creación, sin las limitaciones económicas impuestas por software propietario. Esto no solo fomenta la innovación, sino que también promueve un desarrollo más humano y sostenible, alineado con valores éticos y estéticos. Además, el intercambio libre de ideas y la colaboración que posibilita el software libre son fundamentales para avanzar hacia un diseño más inclusivo y democrático. Con estas herramientas, los profesionales pueden explorar nuevas formas de creación sin depender de costosos programas privativos, lo que abre el campo a una mayor diversidad de enfoques y soluciones creativas.

¿Como emplear el software libre en la arquitectura y el diseño?

Alternativa para	Software libre
• Navegadores web (Internet Explorer)	• Epiphany, Nautilus Firefox
• Clientes Correo (Outlook, Procesadores de texto (MS Office)	• Thunderbird, kmail
Alternativas para	**Software libre**
• Adobe, Photoshop, Paint	Gimp, kpaint, image Magick
Alternativa para	**Software libre**
• Trabajo con CAD/ CAM y CAE (AutoCad)	• PhytonCad, Qcad

USO DE PATRONES DE DISEÑO: UN CASO PRÁCTICO

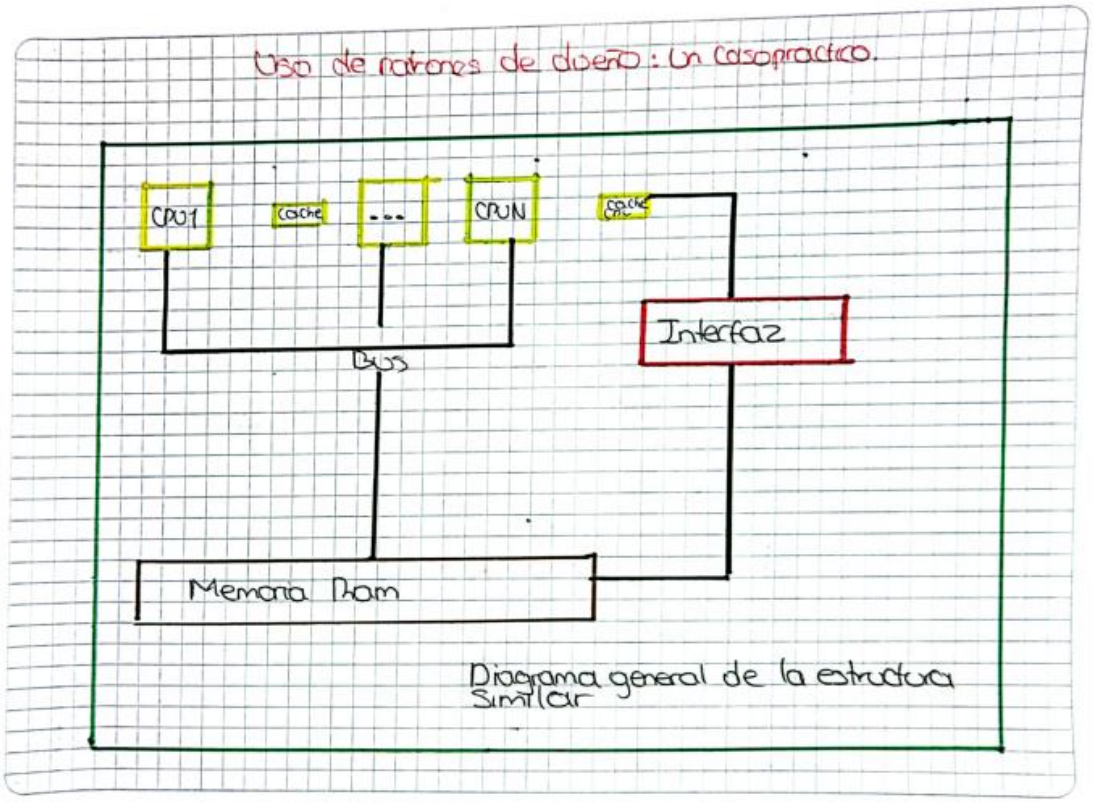
Resumen

Los patrones de diseño son soluciones comprobadas a problemas recurrentes en el desarrollo de software y se consideran fundamentales en la formación de ingenieros y científicos de la computación. Según Gómez, Jiménez y Arroyo (2009), el conocimiento de estos patrones debería formar parte de la educación básica de los estudiantes en ciencias de la computación, especialmente en la programación orientada a objetos. En el curso de Ingeniería de Software de la Escuela de Ciencias de la Computación e Informática, los estudiantes desarrollan aplicaciones web utilizando patrones de diseño. Sin embargo, para aplicar una mayor cantidad de patrones, se aprovechó un proyecto del curso de Arquitectura de Computadoras, que consistió en simular una computadora funcional con componentes como la CPU, la memoria RAM y el bus de comunicación entre estos. En el artículo, se presenta un marco teórico, la descripción del proyecto, la metodología seguida para aplicar los patrones y los resultados obtenidos, así como las conclusiones y los trabajos futuros posibles.

Reflexión

El uso de patrones de diseño en proyectos académicos no solo optimiza el desarrollo de software, sino que también proporciona a los estudiantes herramientas valiosas para enfrentar problemas complejos de manera estructurada y eficiente. Integrar estos patrones en el currículo universitario, como sugiere la investigación, fomenta una comprensión profunda de buenas prácticas en programación, algo esencial para los futuros profesionales del área. Además, al aplicar patrones en contextos más amplios, como el proyecto de simulación de una computadora, los estudiantes experimentan cómo estas soluciones pueden escalar y adaptarse a diferentes tipos de sistemas. Esto no solo mejora su capacidad técnica, sino que también refuerza su creatividad y capacidad de

adaptación. Con un enfoque práctico y continuo, los patrones de diseño pueden transformar la manera en que los estudiantes abordan el desarrollo de software, permitiéndoles crear soluciones más robustas y fácilmente mantenibles.



Patrones de diseño y multiculturalidad

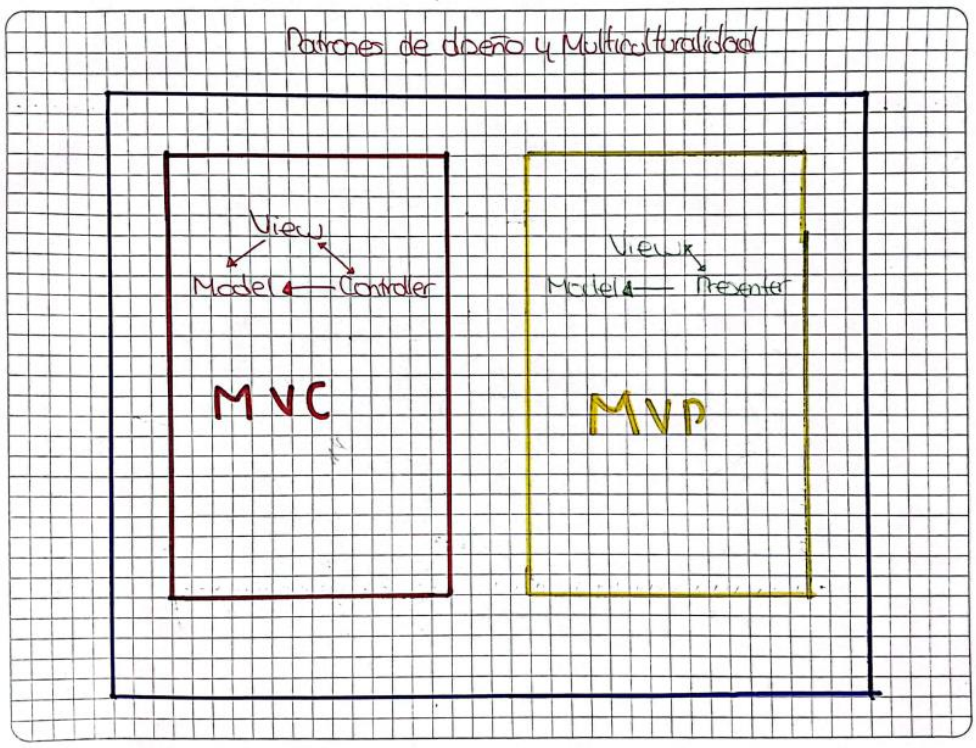
Resumen

En la era de la información, impulsada por las Tecnologías de la Información y la Comunicación (TICs), muchos aspectos de la vida humana han cambiado, incluyendo el comercio, la educación y el turismo. Este cambio también ha influido en los patrones culturales de las sociedades, promoviendo la multiculturalidad. En este contexto, la disciplina de Interacción Humano-Computador (HCI) se enfoca en mejorar la relación entre las personas y los sistemas informáticos, buscando crear interfaces efectivas que respeten la diversidad cultural. Los patrones de diseño, inicialmente propuestos por Alexander en 1977 en el campo de la arquitectura, han sido adaptados en HCI para desarrollar aplicaciones web inclusivas. Estos patrones ayudan a los diseñadores a resolver problemas recurrentes en el desarrollo de software. El artículo destaca la importancia de integrar lineamientos multiculturales en el diseño web para garantizar que los usuarios de diferentes contextos geográficos y culturales experimenten interfaces que les resulten comprensibles y efectivas. Al incluir principios multiculturales en los patrones de diseño, se busca mejorar la interacción y satisfacción del usuario.

Reflexión

La multiculturalidad en el diseño web no es solo una necesidad técnica, sino una cuestión ética y social. A medida que la globalización sigue expandiéndose, los diseñadores de software deben adaptarse a una diversidad de culturas, valores y perspectivas, lo que requiere que los sistemas interactivos sean inclusivos y respetuosos de esa diversidad. La integración de patrones de diseño que reflejen principios multiculturales es esencial para mejorar la accesibilidad y efectividad

de las aplicaciones, especialmente cuando se desarrollan para usuarios globales. Este enfoque no solo mejora la calidad de la interfaz, sino que también contribuye a una experiencia de usuario más enriquecedora y universal. De igual manera, promover un lenguaje común entre los diseñadores de software, basado en patrones de diseño, facilita la creación de soluciones más cohesivas y funcionales. En última instancia, la implementación de estos principios puede hacer que las aplicaciones web no solo sean técnicamente eficientes, sino también culturalmente sensibles, lo que incrementa la satisfacción y fidelidad del usuario, independientemente de su ubicación.



La Arquitectura de Software en el Proceso de Desarrollo: Integrando MDA al Ciclo de Vida en Espiral

Resumen

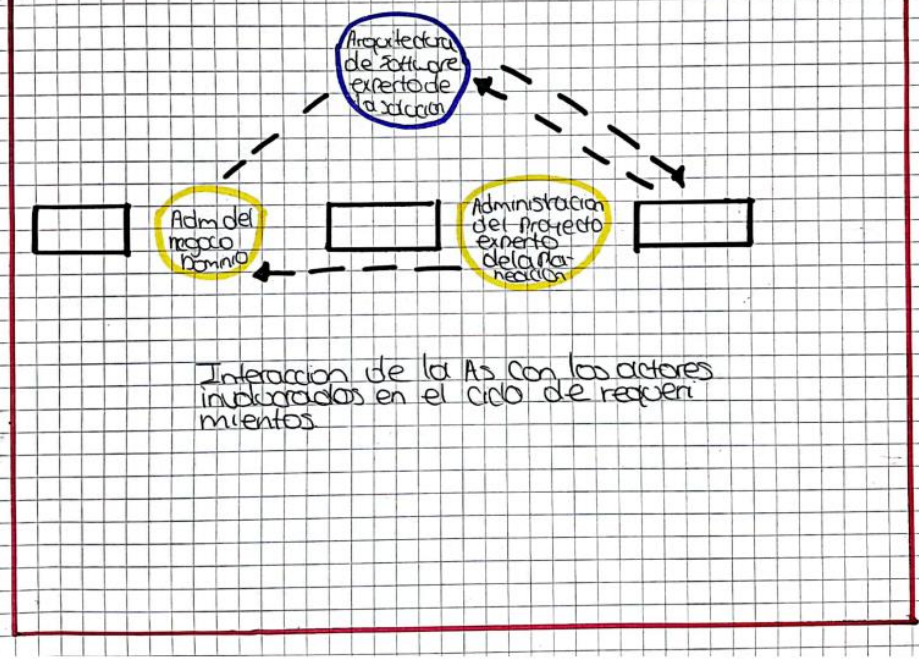
La arquitectura dirigida por modelos (MDA) propuesta por la OMG se basa en el uso de modelos en diferentes niveles de abstracción (CIM, PIM, PSM) para guiar el desarrollo de software y generar automáticamente código desde esos modelos. Este enfoque promueve la transformación de modelos como mecanismo para lograr este paso entre niveles. El modelo en espiral, por su parte, es un enfoque evolutivo que incluye análisis de riesgos en sus fases de planificación, ingeniería y evaluación del cliente. El artículo aborda dos objetivos principales: analizar el grado de participación de la arquitectura del software en el ciclo de vida en espiral y evaluar el impacto de MDA en los riesgos asociados. Uno de los problemas del ciclo de vida tradicional es la pérdida de trazabilidad entre artefactos generados en distintas etapas, lo que MDA soluciona mediante transformaciones automáticas entre modelos, garantizando la trazabilidad y mejorando la gestión de riesgos. Además, MDA facilita la adaptación a cambios tecnológicos sin afectar los modelos de alto nivel.

Reflexión

El enfoque de MDA, al permitir una transformación continua de modelos y generar código automáticamente, ofrece una gran ventaja en la gestión de riesgos y cambios durante el desarrollo del software. Uno de los mayores retos en el desarrollo de software es la gestión de la trazabilidad y la coherencia entre las distintas fases del ciclo de vida, especialmente cuando surgen cambios en los requisitos o el diseño. MDA minimiza este problema, garantizando que los modelos se mantengan actualizados y alineados con los cambios sin perder la visión global del sistema. Además, la separación de los aspectos tecnológicos permite una mayor flexibilidad

para adaptarse a nuevos requisitos sin que se vea afectada la estructura del software, lo que mejora la eficiencia y reduce el riesgo de fallos. Sin embargo, la implementación de MDA requiere una comprensión profunda de las herramientas y metodologías involucradas, lo que puede representar un desafío para los equipos de desarrollo. A largo plazo, este enfoque puede transformar cómo se gestionan los proyectos de software, haciendo el proceso más ágil, flexible y resistente a los cambios.

La arquitectura de Software en el proceso de desarrollo: Integrando MDA al ciclo de vida Espiral.



Interacción de la AS con los actores involucrados en el ciclo de requerimientos

Evaluación de una Arquitectura de Software

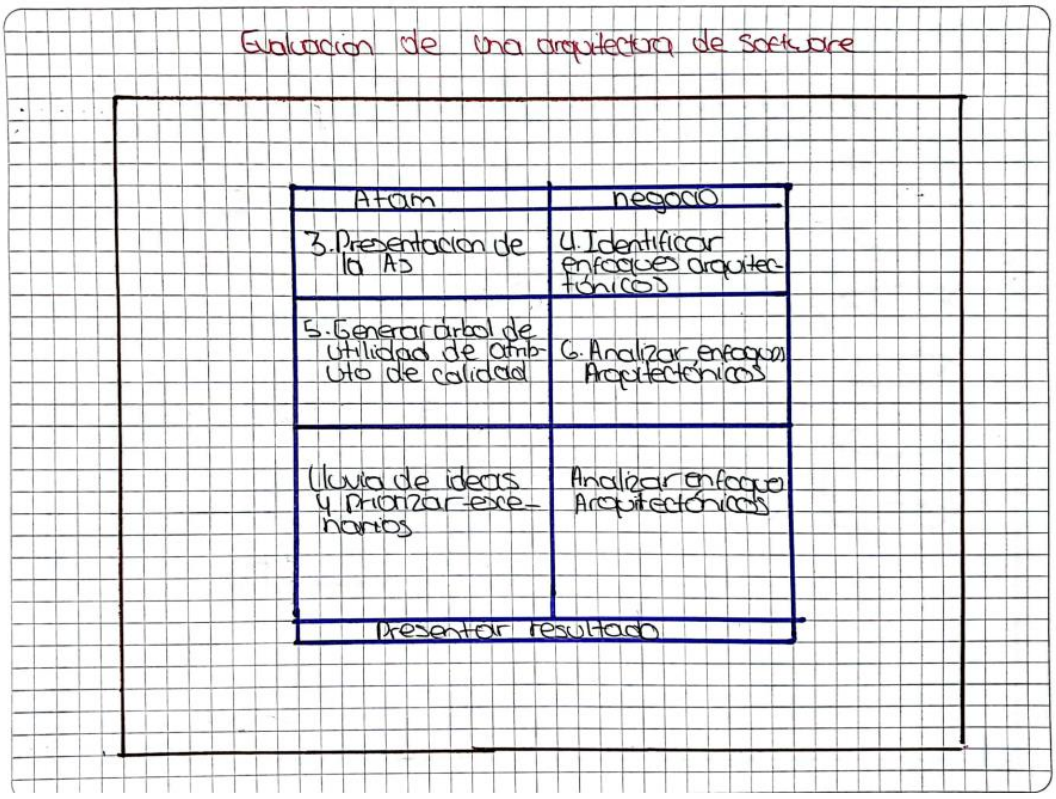
Resumen

El sistema de evaluación y calificación de las Direcciones Territoriales de Salud, las EPS y las IPS en Colombia está regulado por la Resolución 4505 de 2012, que establece la estructura del reporte de actividades en salud pública. La Secretaría Municipal de Salud de Villavicencio (SMSV) recibe reportes periódicos de las IPS con información sobre la población no asegurada, lo que representa un alto volumen de datos, generando errores en los registros. La SMSV valida manualmente los archivos, lo que no garantiza la integridad de la información. Para solucionar esto, se requiere un aplicativo que valide y reporte errores en los archivos, diferenciando entre errores estructurales y lógicos, mejorando la calidad de los informes. Además, se requiere generar alarmas para el seguimiento de citas y procedimientos relacionados con enfermedades de alto impacto. El uso de un Sprint cero permitió evaluar la arquitectura del software para el sistema, y la implementación del patrón MVC asegura la flexibilidad y adaptabilidad ante futuros cambios. Este sistema podría ser replicado en otras secretarías de salud con ajustes específicos.

Reflexión

El desarrollo de un sistema automatizado para la validación de datos en salud es fundamental en un contexto en el que el manejo de grandes volúmenes de información es crucial para tomar decisiones acertadas. La manualidad en el proceso de validación, como lo que ocurre en la SMSV, no solo ralentiza el flujo de trabajo, sino que también pone en riesgo la precisión de la información, afectando decisiones importantes para la salud pública. Implementar un sistema que diferencie y reporte los errores de manera eficiente no solo mejoraría la calidad de los datos, sino también la confianza en las decisiones basadas en esos informes. Además, la flexibilidad del sistema, gracias a la arquitectura MVC y el Sprint cero, garantiza que el software pueda ajustarse a futuros cambios sin perder

funcionalidad. Este enfoque permite que no solo la SMSV, sino otras entidades de salud, se beneficien de una herramienta adaptable y efectiva. A largo plazo, esto optimizaría el uso de los recursos y mejoraría la eficiencia en la atención de la población, destacando la importancia de la innovación tecnológica en la administración pública de salud.



Marco de referencia de arquitectura de software para aplicaciones web y móviles

Resumen

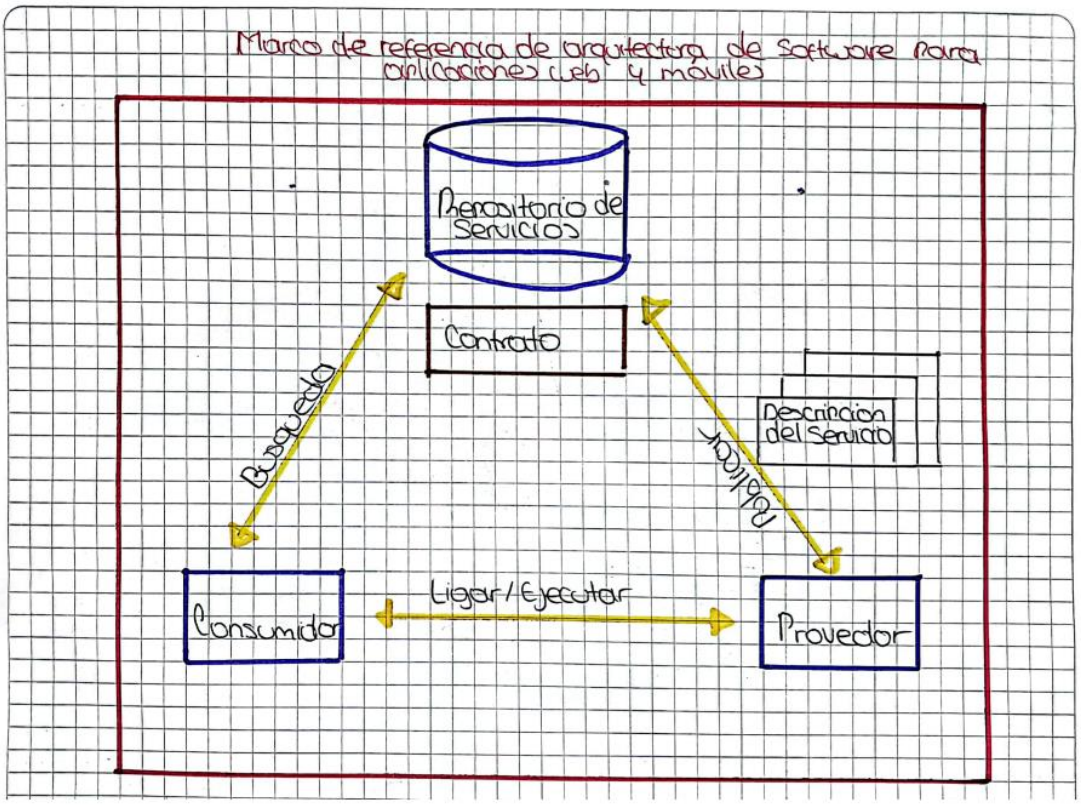
La movilidad en el ámbito de las aplicaciones corporativas es una necesidad, pero también representa un reto debido a sus costos. Para abordar esta cuestión, se propone una plataforma que gestione el desarrollo de la interfaz y la integración con aplicaciones internas de manera segura y escalable. La arquitectura Cliente/Servidor, combinada con la programación orientada a objetos (POO) y la metodología de capas con el patrón Modelo Vista Controlador (MVC), es clave para conseguir este objetivo. Además, se destaca la importancia de herramientas de software libre y código abierto en el desarrollo de aplicaciones web y móviles, ya que permiten el ahorro de costos en licencias y servidores. El uso de estas herramientas facilita la creación de aplicaciones modulares, livianas, portables y escalables. La optimización de recursos en red y bases de datos mejora los tiempos de respuesta, y el uso de controles de interfaz, como ExtJS y Sencha Touch, ofrece soluciones fáciles de personalizar y usar.

Reflexión

El desarrollo de aplicaciones web y móviles, especialmente en entornos corporativos, debe centrarse no solo en la funcionalidad y la integración, sino también en la optimización de los recursos y la escalabilidad. La arquitectura Cliente/Servidor y el uso de patrones como MVC permiten estructurar el software de manera eficiente, facilitando tanto el desarrollo como el mantenimiento a largo plazo. Además, el aprovechamiento de software libre y de código abierto no solo reduce los costos, sino que también fomenta la flexibilidad y la innovación, permitiendo que las soluciones se adapten a las necesidades cambiantes del entorno tecnológico. La modularidad y portabilidad de las aplicaciones, junto con la mejora de los tiempos de respuesta, son aspectos clave en la creación de sistemas eficientes y de alto rendimiento. En un mundo cada vez más interconectado y móvil,

garantizar la seguridad y la optimización de los recursos es fundamental para ofrecer aplicaciones de calidad que puedan escalar conforme crecen las demandas del negocio. El uso adecuado de estas tecnologías puede ser un factor diferenciador para el éxito a largo plazo de las empresas.

Marco de referencia de arquitectura de Software para aplicaciones Web y móviles



Arquitectura de software Culti Ventura, herramienta de soporte a la enseñanza-aprendizaje de la cultura Moche “Videojuegos y realidad humana”

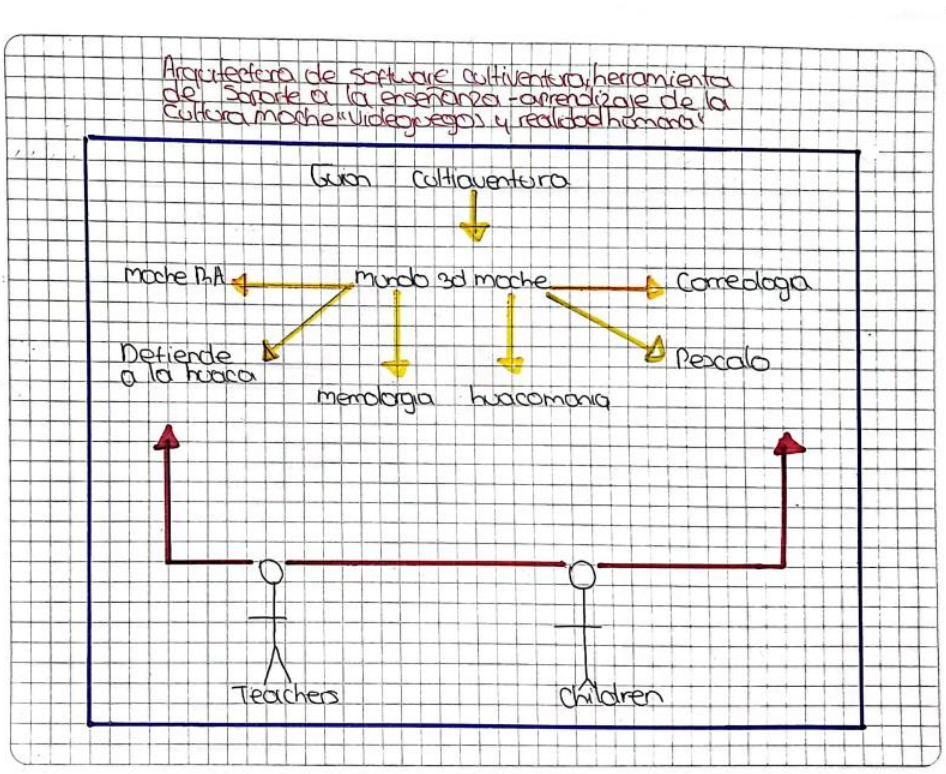
Resumen

El artículo describe la arquitectura de software utilizada en el desarrollo de Cultiventura, una herramienta educativa que apoya la formación de la identidad cultural en el proceso de enseñanza-aprendizaje de la cultura Moche. Utilizando tecnologías de videojuego y realidad aumentada, el software busca acercar a los estudiantes a la cultura peruana de manera divertida e interactiva. El diseño de la arquitectura se realizó en un proceso ágil, permitiendo una evolución constante del producto, ajustando la solución a los requerimientos funcionales y no funcionales. La arquitectura facilita la integración de conceptos culturales y la creación de recursos interactivos que favorecen el aprendizaje. Además, el proyecto responde a las políticas educativas del Ministerio de Cultura y Educación de Perú, que promueven la diversidad cultural. El modelo de desarrollo ágil permite gestionar los componentes del sistema de forma eficiente, asegurando la calidad y reducción de riesgos en el proceso.

Reflexión

El desarrollo de herramientas educativas como Culti Ventura demuestra cómo la tecnología, en particular los videojuegos y la realidad aumentada, puede desempeñar un papel crucial en la preservación y transmisión de la identidad cultural. Al integrar elementos culturales de una manera interactiva, este tipo de aplicaciones no solo fomentan el aprendizaje, sino que también contribuyen a la revalorización de las culturas locales, en este caso, la cultura Moche. El uso de un enfoque ágil permite una adaptación rápida a las necesidades cambiantes y facilita la evolución del producto para ajustarse a las expectativas del usuario final. Este modelo de desarrollo es especialmente relevante en el ámbito educativo, donde la interacción y la motivación son clave para el aprendizaje significativo. Además, la sistematización de los elementos del software, en especial en un contexto educativo, ayuda a garantizar que los recursos sean accesibles, funcionales y de calidad, promoviendo una experiencia de aprendizaje eficaz. Cultiventura es un ejemplo claro de

cómo la tecnología puede ser utilizada para enriquecer el proceso educativo, además de cumplir un rol fundamental en la preservación de las tradiciones culturales.



Arquitectura de Software para Sistema Gestión de Inventarios.

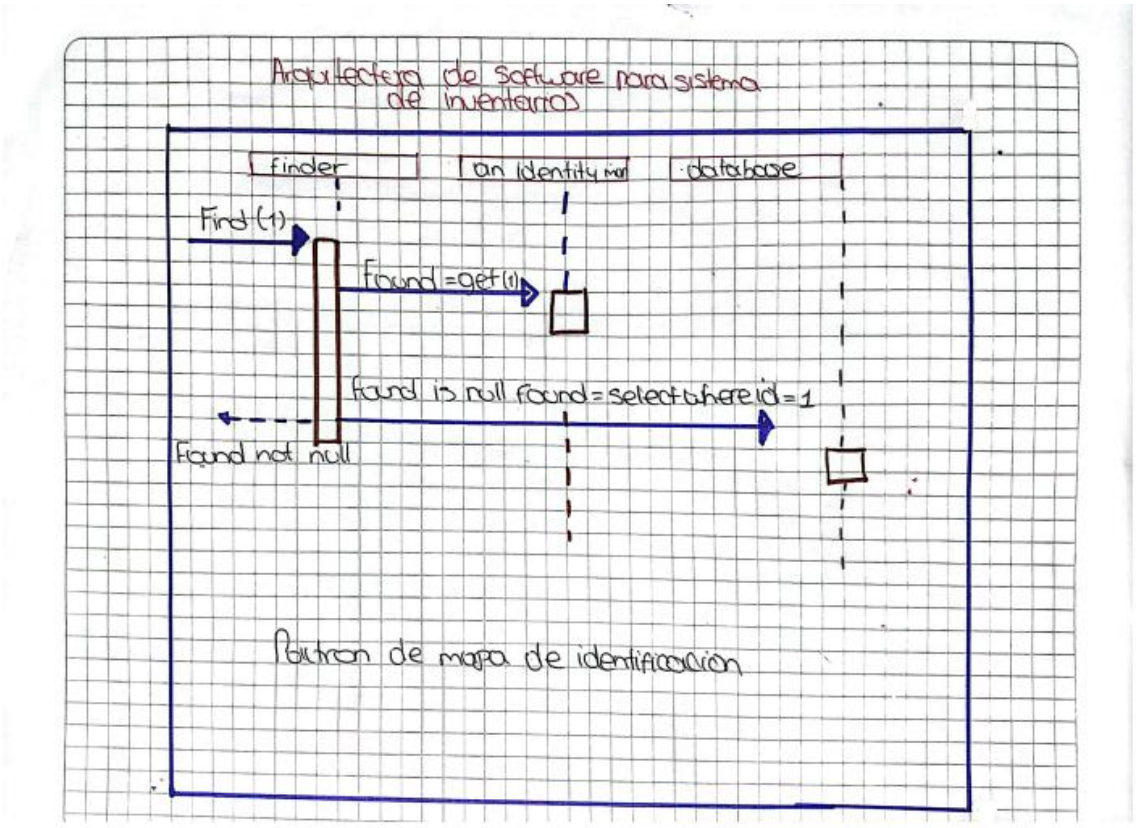
Resumen

El trabajo presenta el diseño de la arquitectura de software para un Sistema de Gestión de Inventario y Almacén. Se analiza la organización estructural del sistema, incluyendo sus componentes, conectores y restricciones. La arquitectura debe ser modular, flexible y satisfacer los requisitos de funcionalidad, eficiencia y confiabilidad. En el contexto de la economía cubana, que enfrenta desafíos tecnológicos, el sistema busca automatizar los procesos de control de inventarios y mejorar la eficiencia empresarial. El proyecto propone un sistema estándar y parametrizable, capaz de adaptarse a las necesidades de los clientes. La metodología de desarrollo RUP se utilizó para definir los principales estilos arquitectónicos, patrones y tecnologías a emplear. La arquitectura propuesta cumple con los requisitos funcionales y no funcionales, asegurando que el sistema sea construible, verificable y administrable. Además, se enfoca en facilitar la reutilización de componentes y mejorar la dinámica de las empresas cubanas.

Reflexión

La arquitectura de software es un componente esencial en el éxito de cualquier sistema, ya que define la estructura básica sobre la que se desarrollará el producto final. En este caso, el desarrollo de un sistema de gestión de inventarios en un contexto económico complejo como el de Cuba resalta la importancia de una planificación detallada y una selección adecuada de tecnologías y metodologías. La necesidad de adaptar los sistemas existentes a los nuevos requisitos refleja cómo la tecnología debe evolucionar para responder a los cambios sociales y económicos. El uso de un enfoque ágil y modular en la arquitectura permite una mayor flexibilidad y adaptación a los cambios, además de facilitar la reutilización de componentes. El proceso de toma de decisiones arquitectónicas es crucial, pues impacta directamente en la calidad, escalabilidad y eficiencia del sistema. Este enfoque también demuestra cómo las decisiones tecnológicas pueden contribuir al perfeccionamiento de procesos y a la mejora de la competitividad de las empresas,

en especial cuando se busca mejorar la transparencia y el control en la gestión de recursos. En definitiva, un diseño arquitectónico bien fundamentado es clave para enfrentar los desafíos tecnológicos y económicos en entornos dinámicos.



Introducción a los Patrones de Diseño

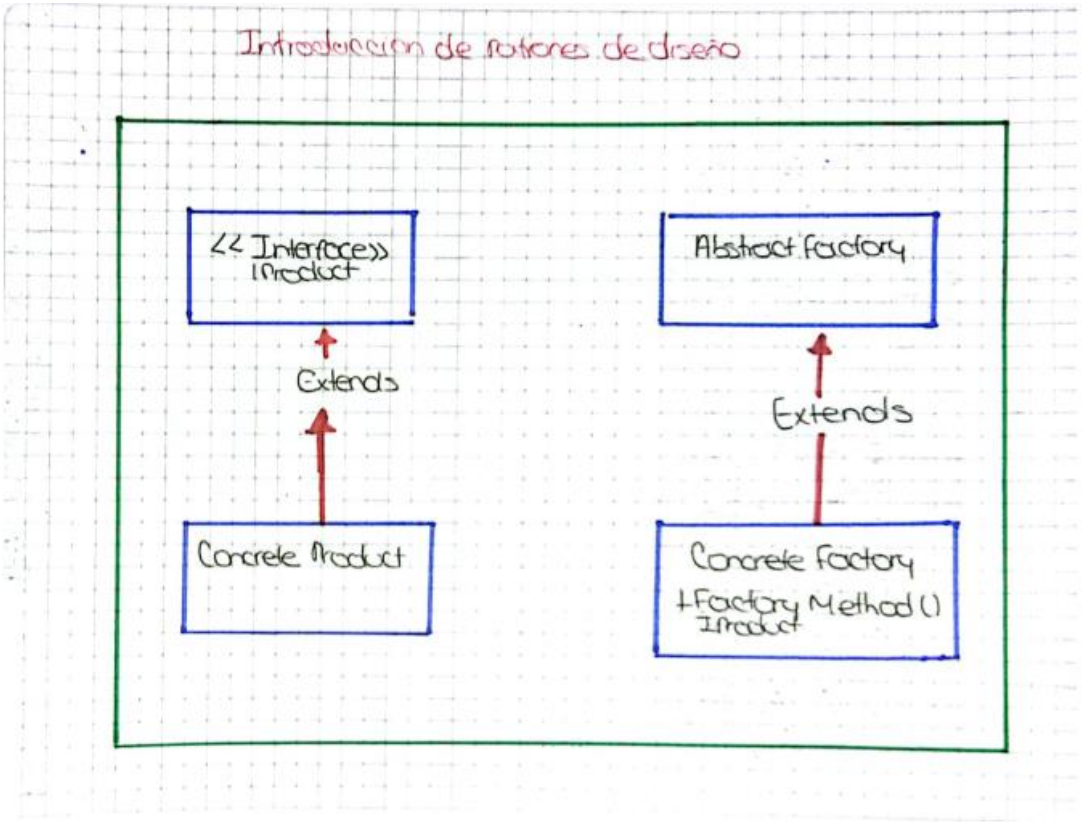
Resumen

Los patrones de diseño tienen su origen en la arquitectura de edificios, iniciados por Christopher Alexander en su libro *Timeless Way of Building* en 1979, donde presentó patrones como soluciones a problemas recurrentes en el diseño arquitectónico. Más tarde, en *A Pattern Language*, formalizó estos patrones para ser aplicados en situaciones de diseño. Estos patrones no son abstractos ni específicos de una cultura o situación, sino que proporcionan soluciones aplicables en diversos contextos. En 1987, Ward Cunningham y Kent Beck adoptaron estos conceptos para la programación orientada a objetos, creando patrones de interacción hombre-máquina. Sin embargo, fue en los años 90 cuando los patrones de diseño tuvieron su mayor impacto en informática con el libro *Design Patterns* de los autores conocidos como *Gang of Four* (GoF), donde se presentaron 23 patrones comunes. Los patrones de diseño ofrecen soluciones reutilizables a problemas recurrentes, lo que facilita su implementación en diversos contextos y su aplicación en distintos sistemas.

Reflexión

La introducción de los patrones de diseño en el ámbito del software, inspirados en la arquitectura, muestra cómo los principios de construcción de estructuras pueden trasladarse a la ingeniería de software, mejorando la organización y reutilización de soluciones. Los patrones no sólo resuelven problemas específicos, sino que también facilitan la comprensión y la flexibilidad del código, permitiendo a los desarrolladores enfrentarse a desafíos de manera más eficiente. El uso de patrones de diseño también fomenta la creación de software más modular, escalable y fácil de mantener, lo que optimiza los procesos de desarrollo. La idea de reutilización de soluciones probadas y efectivas asegura que los desarrolladores no tengan que "reinventar la rueda" cada vez, lo cual mejora la productividad y calidad del software. Además, la capacidad de extender un patrón para adaptarse a nuevos requerimientos, como en el caso de la conexión a diferentes bases de datos,

demuestra la versatilidad y adaptabilidad de los patrones en el desarrollo de software. Esta capacidad de ampliar y personalizar las soluciones a través de patrones abre un abanico de posibilidades para resolver problemas de diferentes naturalezas. El desarrollo basado en patrones también refuerza el concepto de buena arquitectura de software, que permite que los sistemas sean más robustos, fáciles de modificar y con mayor vida útil. En definitiva, los patrones de diseño proporcionan un marco de trabajo probado y eficiente que beneficia tanto a los desarrolladores como a las organizaciones al facilitar la creación de software de calidad.



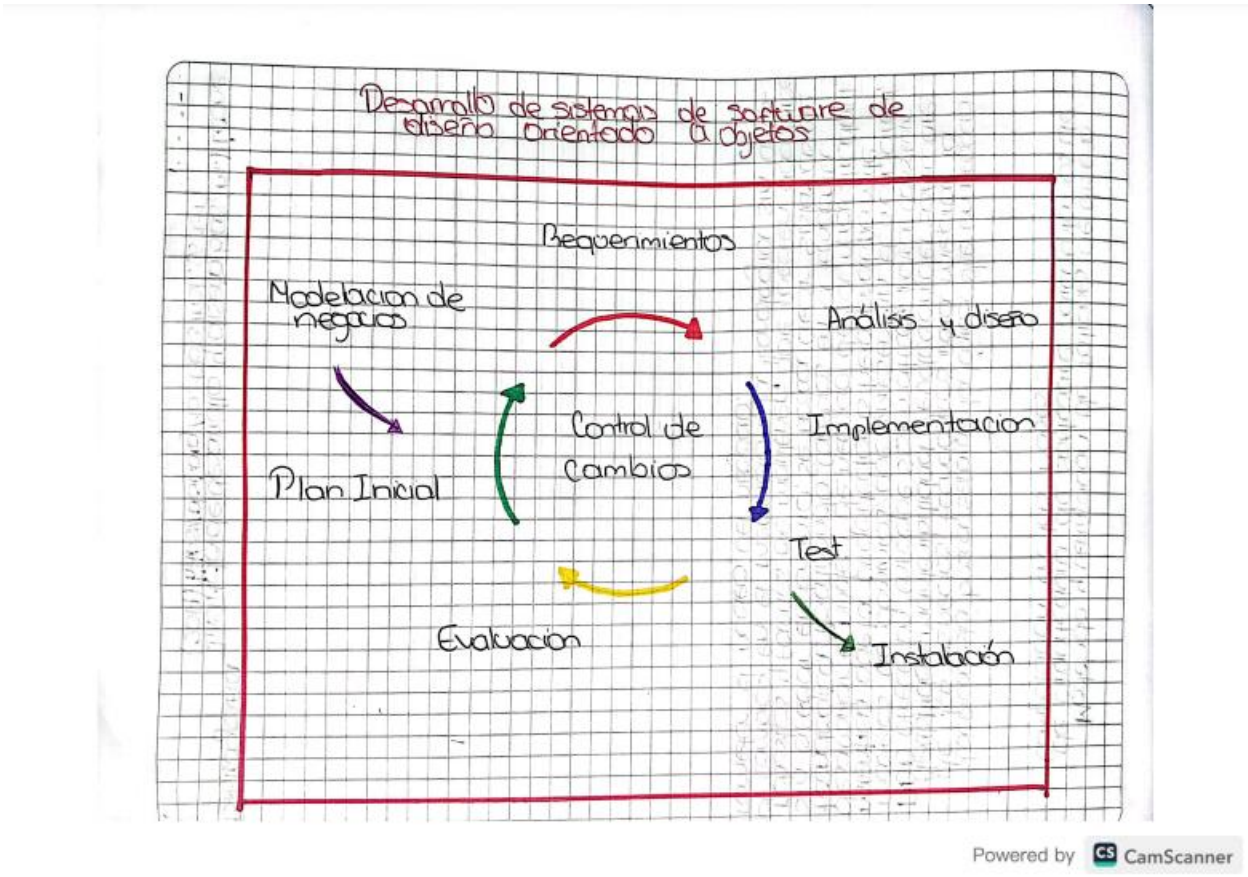
Desarrollo de sistemas de software con patrones de diseño orientado a objetos

Resumen

Aborda la implementación de patrones de diseño orientados a objetos en la creación de un sistema de software denominado "Intranet Industrial", desarrollado en la Facultad de Ingeniería Industrial de la Universidad Nacional Mayor de San Marcos. Se analiza el impacto económico de la adopción de estos patrones en comparación con un enfoque que no los considera. En la primera sección, se introducen conceptos teóricos y se clasifican los patrones más reconocidos en el ámbito industrial. A continuación, se detallan los patrones aplicados en el desarrollo del sistema y se presentan de manera formal. La sección final ofrece una descripción del sistema, sus módulos y herramientas, así como una estimación de costos utilizando el patrón "Informador" y el método COCOMO. Se concluye que la implementación de patrones contribuye a mejorar la eficiencia y a disminuir los costos en el desarrollo de software.

Reflexión

El enfoque de utilizar patrones de diseño en el desarrollo de software es una estrategia altamente efectiva que no solo optimiza el proceso de creación, sino que también promueve la reutilización y la escalabilidad. Este trabajo es un excelente ejemplo de cómo integrar teoría y práctica, mostrando que la implementación de patrones no solo es beneficiosa desde el punto de vista técnico, sino que también tiene repercusiones positivas en la gestión de costos. La investigación resalta la importancia de adoptar estándares en el desarrollo de sistemas complejos, ofreciendo un modelo que puede servir de referencia para futuros proyectos en el ámbito académico e industrial.



Bibliografía

1. López, D., & Maya, E. (s/f). *Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web*. Surl.li. Recuperado el 21 de noviembre de 2024, de http://surl.li/adpdwe

2. INGAR - Instituto de Desarrollo y Diseño, Blas, M. J., Leone, H., INGAR - Instituto de Desarrollo y Diseño, Gonnet, S., & INGAR - Instituto de Desarrollo y Diseño. (2019). 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*, *35*(35), 1–17. https://doi.org/10.17013/risti.35.1-17

3. *Análisis comparativo de Patrones de Diseño de Software*. (n.d.). Unirioja.Es. Retrieved November 21, 2024, from https://dialnet.unirioja.es/servlet/articulo?codigo=9042927

4. *Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web*. (n.d.). Scielo.Cl. Retrieved November 21, 2024, from https://www.scielo.cl/scielo.php?pid=S0718-07642013000300012&script=sci_arttext&tlng=en

5. *Arquitectura de software, esquemas y servicios*. (n.d.). Unirioja.Es. Retrieved November 21, 2024, from https://dialnet.unirioja.es/servlet/articulo?codigo=4786655

6. Trabajo de Diploma Para Optar Por el Título, de I. en C. I. (n.d.). *Arquitectura de software para videojuegos desarrollados sobre el motor de juego Unity 3D*. Uci.Cu. Retrieved November 21, 2024, from https://repositorio.uci.cu/jspui/bitstream/123456789/9382/1/TD_08978_17.pdf

7. Guerrero, L. (2008). Arquitectura de Software orientada a la creación de mricomundos para la enseñanza y el aprendizaje. *Avances Investigación en Ingeniería*, *1*(8), 88–95. https://revistas.unilibre.edu.co/index.php/avances/article/view/2637

8. Ferrandis Homsi, A. (2021). *Desarrollo de una herramienta para el aprendizaje de patrones de diseño software*. Universitat Politècnica de València.

9. Palmero, M. A. S., Martínez, N. S., & Grass, O. Y. R. (2019). Revisión de elementos conceptuales para la representación de las arquitecturas de referencias de

software. *Revista Cubana de Ciencias Informáticas*, *13*(1), 143–157. http://scielo.sld.cu/scielo.php?pid=S2227-18992019000100143&script=sci_arttext

10. Bastarrica, M. C. (n.d.). *Atributos de Calidad y Arquitectura del Software*. Upm.Es. Retrieved November 21, 2024, from http://www.grise.upm.es/rearviewmirror/conferencias/jiisic04/Tutoriales/tu4.pdf

11. De la, C. de I. (n.d.). *SABER. Revista Multidisciplinaria del*. Redalyc.org. Retrieved November 21, 2024, from https://www.redalyc.org/pdf/4277/427739438009.pdf

12. Cambarieri, M. G., Difabio, F., & Martínez, N. G. (n.d.). *Implementación de una Arquitectura de Software guiada por el Dominio*. Edu.Ar. Retrieved November 21, 2024, from https://sedici.unlp.edu.ar/bitstream/handle/10915/115198/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y

13. Parnas, D. L. (1972). On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, *15*(12), 1053–1058. https://doi.org/10.1145/361598.361623

14. *Desarrollo de una arquitectura de software para el robot móvil Lázaro*. (n.d.). Scielo.Cl. Retrieved November 21, 2024, from https://www.scielo.cl/scielo.php?pid=S0718-33052018000300376&script=sci_arttext

15. with:, P. in C. (n.d.). *Arquitectura de software académica para la comprensión del desarrollo de software en capas*. Econstor.Eu. Retrieved November 21, 2024, from https://www.econstor.eu/bitstream/10419/130825/1/837816424.pdf

16. Calderón Castro, A. (2003). Lenguajes de patrones de diseño de software bajo una perspectiva cognoscitivista. *InterSedes*, *IV*(7), 167–189. https://www.redalyc.org/articulo.oa?id=66640712

17. Luna-García, H., Mendoza-González, R., & Álvarez-Rodríguez, F.-J. (2015). Patrones de diseño para mejorar la accesibilidad y uso de aplicaciones sociales para

adultos mayores. *Comunicar*, *XXII*(45), 85–94. https://www.redalyc.org/articulo.oa?id=15839609009

18. Lagunes García, G., López Martínez, I., Peláez Camarena, G. S., Abud Figueroa, M. A., & Olivares Zepahua, B. A. (2015). Arquitectura de software de una aplicación móvil para desarrollar un sistema de identificación por radiofrecuencia. *ReCIBE. Revista Electrónica de Computación, Informática, Biomédica y Electrónica*, *1*. https://www.redalyc.org/articulo.oa?id=512251501004

19. *Scientia Et Technica*. (n.d.). Redalyc.org. Retrieved November 21, 2024, from https://www.redalyc.org/pdf/849/84933912003.pdf

20. *Desarrollo de sistemas de software con patrones de diseño orientado a objetos*. (n.d.). Edu.Pe. Retrieved November 21, 2024, from https://cybertesis.unmsm.edu.pe/backend/api/core/bitstreams/92b01647-3651-4e75-9fd1-b00cb53826b1/content

21.

22. Cárdenas-Gutiérrez, J. A., Barrientos-Monsalve, E. J., & Molina-Salazar, L. (2022). Arquitectura de Software para el desarrollo de herramienta Tecnológica de Costos, Presupuestos y Programación de obra. *I+D Revista de Investigaciones*, *17*(1), 85–95. https://doi.org/10.33304/revinv.v17n1-2022007

23. Ospina Torres, M. H., & Luna, C. L. (2013). Una arquitectura basada en software libre para archivos web. *Enl@ce*, *10*(1), 53–72. https://www.redalyc.org/articulo.oa?id=82326270005

24. Pérez-Sotelo, J. S., Cuesta, C. E., & Ossowski, S. (2011). Patrones de adaptación para arquitecturas de software basadas en tecnologías del acuerdo. *REICIS. Revista Española de Innovación, Calidad e Ingeniería Del Software*, *7*(3), 6–25. https://www.redalyc.org/articulo.oa?id=92222551003

25. Crespo, V. (2007). CÓMO EMPLEAR EL SOFTWARE LIBRE EN LA ARQUITECTURA Y EL DISEÑO. *AU. Arquitectura y Urbanismo*, *XXVIII*(2), 92–95. https://www.redalyc.org/articulo.oa?id=376839852016

26. Salazar Bermúdez, G., Montenegro Jiménez, I., & Rodríguez Rodríguez, L. (2013). USO DE PATRONES DE DISEÑO: UN CASO PRÁCTICO. *Revista Ingeniería*, *22*(2), 45–59. https://doi.org/10.15517/ring.v22i2.8220

27. González, I. (2012). Patrones de diseño y milticulturalidad. *Prisma Tecnológico*, *3*(1), 31–34. https://revistas.utp.ac.pa/index.php/prisma/article/view/543

28. *La Arquitectura de Software en el Proceso de Desarrollo: Integrando MDA al Ciclo de Vida en Espiral*. (n.d.). Edu.Ar. Retrieved November 21, 2024, from https://revistas.unla.edu.ar/software/article/view/103

29. Agudelo, O., Sanabria, F. R., & Rodríguez, S. V. (2021). Evaluación de una Arquitectura de Software. *Prospectiva*, *19*(2). https://doi.org/10.15665/rp.v19i2.2636

30. Babahoyo, I. T. S., Maliza Martinez, C. A., López Mendizábal, V. L., Babahoyo, I. T. S., Mackliff Peñafiel, V. V., & Babahoyo, I. T. S. (2016). Marco de referencia de arquitectura de software para aplicaciones web y móviles. *Journal of Science and Research*, *1*(CITT2016), 72–75. https://doi.org/10.26910/issn.2528-8083vol1isscitt2016.2016pp72-75

31. *"Arquitectura de software Cultiventura, herramienta de soporte a la enseñanza-aprendizaje de la cultura Moche 'Videojuegos y realidad humana.'"* (n.d.). Surl.Li. Retrieved November 21, 2024, from http://surl.li/nuluns

32. *Arquitectura de Software para Sistema Gestion de Inventarios*. (n.d.). Uci.Cu. Retrieved November 21, 2024, from https://repositorio.uci.cu/jspui/handle/ident/TD_0201_07

33. *Introducción a los Patrones de Diseño*. (n.d.). ..N9.Cl. Retrieved November 21, 2024, from https://n9.cl/ti4ix

34. *Desarrollo de sistemas de software con patrones de diseño orientado a objetos*. (n.d.). Edu.Pe. Retrieved November 21, 2024, from https://cybertesis.unmsm.edu.pe/backend/api/core/bitstreams/92b01647-3651-4e75-9fd1-b00cb53826b1/content