RESUMENES DE LA REVISTA

Anyi susey casanova

La edad de oro de la arquitectura de software

Anyi nos habla que la arquitectura de software ha evolucionado donde antes se usaba para describir sistemas de forma general. Ya hoy en día es una guía esencial para el desarrollo web, donde muestra la evolución de creciente complejidad del mundo digital.

Camilo :

Arquitectura de software: fundamentos, teoria y práctica

Camilo habla sobre la importancia de la arquitectura de software en el desarrollo de sistemas, ya que guía su construcción y mantenimiento mediante decisiones clave de diseño. La arquitectura incluye componentes, conectores, configuraciones, características no funcionales y patrones reutilizables que optimizan la ejecución de sistemas. Es especialmente útil para líneas de productos y estilos arquitectónicos.

Carolina:

Arquitectura para herramienta de costos y programación

Habla sobre un programa educativo para costos , presupuesto ye entre otras en una universidad, Esto apoya el aprendizaje autónomo de los estudiantes , utilizando diagramas uml , como el de secuencia, que simula el flujo desde el registro del presupuesto, ayudando a los estudiantes.

CRISTIAN:

Documentación y analisis de principales framework de arquitectura de software en aplicaciones empresariales.

La arquitectura de software es fundamental en el desarrollo de sistemas empresariales eficientes, especialmente para herramientas como ERP y CRM. Este análisis revisa frameworks arquitectónicos destacados:

1. Arquitectura en capas: organiza responsabilidades jerárquicamente.

2. Cliente-servidor: separa roles entre clientes que solicitan datos y servidores que responden.

3. Arquitectura en tres capas: divide funciones en presentación, lógica y datos, logrando.

Ya finalizando comenta que las empresas hay dos sectores, cada uno maneja sus factores, erp que se encarga de la gestión del inventario, y el crm es como la gestión de datos, el marketing, entonces mayoritariamente esos dos grupos de utiliza diferentes arquitectura de software.

CRISTIAN JEANPOOL:

Análisis comparativo de patrones de diseño mvc y mvp para el rendimiento de aplicaciones web.

Habla de la eficiencia de los patrones de diseño MVC (Modelo-Vista-Controlador) y MVP (Modelo-Vista-Presentador) en el desarrollo de aplicaciones web, con el objetivo de determinar cuál es más adecuado. A través de pruebas prácticas, se evaluaron aspectos como la organización del código, la facilidad de mantenimiento y el impacto en el rendimiento.

DYLAN SANTIAGO NARVÁEZ PINTO

Isabella Cordoba:

Ivan:

No respondió

JHON SEBASTIÁN PENNA ARIAS:

Arquitectura hexagonal:

Busca aislar la lógica de negocio del resto del sistema mediante la integración de interfaces externas como bases de datos y APIs. Esta arquitectura divide el sistema en un núcleo central (lógica de negocio) y adaptadores que conectan con el exterior, manteniendo cada parte desacoplada.

PREGUNTA

como ayuda la arquitectura hexagonal a la arquitectura de software

JOSÉ MANUEL GASCA BONILLA

Modelo y herramienta de software

El modelo propone un arquitecto Contreras perspectivas conceptual lógica y física conceptual básicamente define que en relación del proyecto para cotización evaluación y identificar riesgos y gestión de riesgos la perceptiva lógica utiliza programas para modelar clases de clases y relaciones del sistema la perspectiva física define la de la base de datosy modelos ciclo incrementales características se pueden decir sobre estas para cuantificar los riesgos de cada fase del desarrollo.

JUAN PABLO BETANCOURT GÓMEZ

NO ENVIÓ NADA

JULIAN DAVIDAD FIERRO CASANOVA

UNA HERRAMIENTA DE ARQUITECTURA PARA UNA HERRAMIENTA DE PATRONES DE DISEÑO

La arquitectura descrita está diseñada específicamente para integrar patrones de diseño en herramientas de desarrollo de software orientado a objetos. Esta arquitectura permite la creación, manipulación y gestión de patrones como elementos básicos de modelado, lo que optimiza tanto la eficiencia como la reutilización del software, beneficiando directamente a los desarrolladores en el proceso de desarrollo.

La herramienta emplea patrones de diseño como Composite y Observer para facilitar la interacción entre la interfaz gráfica de usuario y la lógica interna del sistema. El patrón Composite estructura elementos en jerarquías de una manera flexible, permitiendo realizar operaciones sobre los objetos de forma más eficiente. Por otro lado, el patrón Observer sincroniza los cambios entre diferentes componentes, asegurando consistencia en todo el sistema.

KEVIN CAMILO MUÑOZ CAMPOS

No respondió

LAURA VALENTINA ARIZA ALEJO

MAPEO EN ARQUITECTURA DE SOFTWARE

Habla sobre e l mapeo en la arquitectura de software se refiere al proceso de analizar y comprender cómo está estructurado internamente un proyecto. Esto implica identificar la organización de módulos, servicios o componentes clave del sistema. Una parte fundamental de este proceso es asegurar que la documentación esté actualizada, ya que la falta de información precisa o la eliminación accidental de datos pueden generar problemas significativos en la gestión del proyecto.

Uno de los principales desafíos del mapeo es cuando la información no está bien organizada o se presenta de manera confusa, lo que dificulta su comprensión y reutilización en otros proyectos. Además, recuperar información perdida no es suficiente si no se presenta de manera ordenada y clara para facilitar su interpretación y comunicación dentro del equipo.

MARIA DEL MAR ANTURRUAGA

Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web

El desarrollo de software en la Coordinación General de Tecnologías de la Información y Comunicación (CGTIC) de la Asamblea Nacional del Ecuador se ha basado en una arquitectura monolítica, lo cual ha generado problemas de mantenimiento, escalabilidad y entrega de aplicaciones web. Este estudio busca analizar tecnologías, metodologías y arquitecturas para adoptar un enfoque basado en microservicios, que permita mejorar estos procesos. Con un enfoque modular y escalable, los microservicios ofrecen una solución eficiente que puede adaptarse mejor a los cambios y requisitos del negocio, promoviendo agilidad y adaptabilidad para cubrir las necesidades actuales y futuras.

MARIA JOSÉ MURCIA

Patrones de usabilidad en la arquitectura de software.

El artículo habla sobre cómo el proyecto STATUS mejora la usabilidad del software desde el principio, en lugar de esperar hasta el final para corregir problemas. Normalmente, la usabilidad se evalúa cuando el sistema ya está terminado, pero STATUS propone incluir opciones útiles como "deshacer", "cancelar" y soporte para varios idiomas desde la etapa de diseño. Esto hace que el software sea más fácil de usar y evita retrabajos costosos. Además, evalúan la usabilidad en cada paso del proceso, haciendo ajustes cuando es necesario. Este enfoque ayuda a los desarrolladores a crear un sistema que sea funcional y agradable para los usuarios desde el inicio.

Mariana Charry

Arquitectura de software para entornos móviles.

Habla sobre un impacto la evolución de los sistemas operativos, porque cada vez son más estables, las aplicaciones móviles ahora son de mayor tamaño y complejidad, lo cual permite a los desarrolladores crearlas.

Se definió una arquitectura móvil, la cual comparte algunos de los principios más reconocidos de la arquitectura de software donde una base más sólida sobre la cual construir software innovador y de alta calidad.

Hay arquitecturas generales y está arquitectura monolítica que es la que se habla en este artículo.

MARIANA GONZALES

Marco de trabajo para seleciionar un patron arquitectónico en el desarrollo de software.

El artículo destaca la relevancia de la arquitectura de software en el desarrollo de proyectos tecnológicos. Presenta un marco de trabajo para seleccionar patrones arquitectónicos que optimicen la calidad, el rendimiento, el mantenimiento y la adaptabilidad del software. Este marco permite a los desarrolladores tomar decisiones basadas en las características del proyecto, como el tipo de desarrollo y los requisitos específicos. Se analizan patrones como MVC, MVP, Microservicios y Arquitectura en la Nube, evaluando su adecuación al contexto y las necesidades del software. La validez del marco se refuerza con la posibilidad de adaptarlo para crear recomendaciones personalizadas.

MAIDY CONDE:

Desarrollo de aplicaciones web utilizando el patron de diseño de modelo/vista/controlador.

El patrón de diseño Modelo/Vista/Controlador (MVC) es útil para organizar sistemas interactivos, pero su implementación en aplicaciones web puede ser compleja debido a la tendencia a particionar durante el diseño. Este documento propone una solución llamada "Partición Flexible de Aplicaciones Web", que permite aplicar el patrón MVC de manera independiente a las particiones. Las aplicaciones se desarrollan y prueban en un único espacio de trabajo, y luego se implementan en diversas arquitecturas sin necesidad de modificar el código.

MAYRA TAMAYO

Análisis comparativo de Patrones de Diseño de Software

Habla sobre la importancia de los patrones de diseño como soluciones estandarizadas para abordar problemas comunes en el desarrollo de software. Resalta su capacidad para evitar la duplicación de código y fomentar la reutilización. Se examinan patrones como Template Method, Modelo-Vista-Controlador (MVC), Modelo-Vista-Presentador (MVP), Modelo Front-Controller y Modelo-Vista-VistaModel (MVVM), evaluando sus componentes, ventajas, desventajas y aplicaciones en distintos contextos. La investigación concluye que no existe un patrón universal superior, ya que cada uno es adecuado para contextos específicos. Los patrones de diseño son fundamentales para mejorar la organización, el mantenimiento y la calidad de los sistemas, ofreciendo una estructura coherente que facilita la evolución y resolución de problemas a largo plazo. Este análisis proporciona a los desarrolladores herramientas para seleccionar el patrón más adecuado según las necesidades del proyecto y para promover prácticas más efectivas y sostenibles.

PATRICIA SARMIENTO:

Marco de trabajo para seleciionar un patrón arquitectónico en el desarroollo de software.

El artículo propone un marco práctico para elegir patrones arquitectónicos en el desarrollo de software, abordando desafíos relacionados con la escalabilidad y la falta de conocimiento técnico que afectan la calidad del producto. Este marco, dividido en cinco etapas, permite identificar patrones clave como MVC, MVP, Microservicios y Arquitectura en la Nube, adaptándolos a las necesidades específicas de cada proyecto. Evalúa aspectos como tiempo, costos, escalabilidad y sostenibilidad, ofreciendo una herramienta útil para orientar a desarrolladores y arquitectos en la selección de patrones óptimos. Este enfoque busca mejorar la estructura del software desde la etapa inicial del proyecto, garantizando mayor mantenibilidad, flexibilidad y menores costos a largo plazo.

VALENTINA SILVA

Análisis Comparativo de Patrones de Diseño de Software

Los patrones de diseño son fundamentales para desarrollar software robusto y adaptable. Cada patrón ofrece una arquitectura específica para satisfacer las necesidades del proyecto. En este análisis, se evalúan patrones reconocidos como Template Method, MVC, MVP, Front Controller y MVVM, considerando su lenguaje, estructura y propósito. Se destacan sus fortalezas y debilidades para que los desarrolladores puedan tomar decisiones informadas, aplicando soluciones más sostenibles y eficientes. Este enfoque busca garantizar que los proyectos de software estén respaldados por arquitecturas flexibles y optimizadas.

WILLIAM

Buenas prácticas en la construcción del software

Este artículo examina diversas arquitecturas y metodologías de desarrollo para crear sistemas de software flexibles y eficientes. Inicialmente, detalla la arquitectura de solución, que orienta el diseño de estructuras integradas adaptadas a necesidades actuales y futuras. También analiza arquitecturas como en capas, monolíticas, microservicios, orientadas a eventos (EDA) y cliente-servidor, que organizan la estructura y comunicación de las aplicaciones, ofreciendo estrategias para optimizar el desarrollo y la implementación de sistemas.

YORDY MUÑOZ

Introducción a los patrones de diseño

Dice que destaca la relevancia de utilizar patrones de diseño como herramientas efectivas para enfrentar desafíos técnicos y mejorar la creación de software. Explica cómo seleccionar un patrón adecuado para cada contexto puede potenciar la flexibilidad, escalabilidad y calidad del código, sin limitar la creatividad del desarrollador. También clasifica los patrones en categorías como creacionales, estructurales y de comportamiento, mostrando su aplicación práctica en lenguajes como UML y programación orientada a objetos (POO). Además, resalta su impacto en la mejora de la durabilidad y reestructuración del software.

RESUMEN DE LOS ARTICULOS DE LA OTRA FICHA

JOHAN CALDERÓN PERDOMO

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

La arquitectura de software organiza sistemas a nivel general, definiendo los elementos y sus relaciones. Su objetivo es facilitar el desarrollo mediante la reutilización de componentes usando arquitecturas de referencia. Este enfoque ha evolucionado incorporando patrones estructurales y escenarios para guiar tanto el diseño como la comprensión de sistemas complejos.

Mayury gonzalez

Monolitos vs. Microservicios en Arquitectura de Software

Aborda cómo las empresas, especialmente aquellas que han trabajado con arquitecturas monolíticas por largo tiempo, enfrentan desafíos al adoptar arquitecturas de microservicios. Las arquitecturas monolíticas agrupan todos los componentes en un único sistema, lo que facilita el desarrollo inicial y ofrece rapidez, pero presentan limitaciones significativas, como dificultades para actualizarse con nuevas tecnologías y un periodo de caducidad más corto en sistemas modernos. Además, cualquier cambio puede requerir una reestructuración completa, lo que complica su mantenimiento.

Por otro lado, la arquitectura de microservicios se enfoca en dividir el sistema en servicios independientes. Esto permite realizar mantenimiento, actualizaciones y cambios de forma aislada, sin afectar al resto del sistema. Aunque su principal ventaja es la flexibilidad y la escalabilidad, su implementación inicial puede ser más compleja debido a la necesidad de herramientas avanzadas, una gestión más robusta de la comunicación entre servicios y mayores recursos tecnológicos.

CARLOS ANDRES

Impacto de implementaciones web del patrón mvc en los requisitos percibidos.

El artículo examina dos enfoques diferentes para implementar el patrón Modelo-Vista-Controlador (MVC) en aplicaciones web, centrándose en cómo afectan atributos de calidad como el tiempo de respuesta y la escalabilidad. Se contrasta la versión tradicional de MVC, basada en un modelo de llamada y respuesta, con otra propuesta que emplea tuberías y filtros implementados mediante coroutines en Python. Aunque la alternativa basada en tuberías presenta un gran potencial, el estudio concluye que necesita más investigación y pruebas para consolidar su uso en aplicaciones prácticas.

Stefany Nikoll

Desarrollo de una arquitectura de software para el robot móvil Lázaro

El artículo describe el desarrollo de una arquitectura de software para el robot móvil Lázaro, estructurada en tres niveles:

1. Gestión de componentes básicos: El primer nivel se encarga de controlar los componentes esenciales del robot.
2. Creación de aplicaciones de control: El segundo nivel ofrece librerías que facilitan el desarrollo de aplicaciones específicas para manejar el robot.
3. Interfaz de usuario y simulador: El tercer nivel incluye una interfaz de usuario con un panel de control y un simulador 3D para programar y monitorear el robot.

JUAN DAVID CERQUERA SALAZAR

Marco de trabajo para seleccionar un patrónn arquitectónico enn el desarrollo de software.

El artículo analiza las arquitecturas de software más utilizadas, destacando sus características y aplicaciones. La arquitectura en la nube sobresale por su seguridad y flexibilidad, mientras que MVC se enfoca en mantenibilidad, rendimiento y eficiencia en memoria. Los microservicios destacan por su seguridad y modularidad, y MVP por su flexibilidad y estabilidad. Además, se identifican los contextos de uso: la nube para aplicaciones web, MVC en dispositivos móviles y de escritorio, microservicios en aplicaciones web y MVP en aplicaciones móviles y web.

ERICK

Arquitectura de software basada en microservicios para desarrollo de aplicaciones web

El artículo aborda el uso de microservicios para implementar una arquitectura de software más flexible y moderna, diseñada para superar las limitaciones de los sistemas monolíticos tradicionales. La investigación destaca cómo los microservicios ofrecen una solución eficiente y adaptable que optimiza el desarrollo y mantenimiento de aplicaciones, permitiendo que los componentes del sistema operen de manera autónoma e independiente.

MARLON

Patrones de diseño(XII): Patrones estructirles -Flyweght

Este artículo explica el patrón de diseño Flyweight, el cual busca optimizar el uso de memoria mediante el intercambio de datos entre objetos similares. Se presentan ejemplos de su implementación en sistemas donde la eficiencia en el manejo de memoria es clave, como en aplicaciones gráficas y videojuegos. Además, se analizan las ventajas y desventajas del patrón Flyweight, junto con las consideraciones importantes a tener en cuenta al aplicarlo.