Exposición de los resúmenes y reflexión de Artículos de la ficha 2694667.

Maria Jose Murcia Martinez

Centro de la Industria, La Empresa y los Servicios.

Análisis y Desarrollo de software

Jesús Ariel Gonzáles Bonilla

Neiva Huila 23 de noviembre de 2024

Desarrolladores:

-Angie Lizeth Trujillo Gonzáles.

No estaba en la clase

-Anyi Zujey Gómez Casanova.

Artículo 11:

La edad de oro de la arquitectura de software.

La arquitectura de software ha evolucionado desde los años 80, pasando de ser una simple descripción de sistemas a convertirse en una disciplina esencial para diseñar y construir software complejo. Hoy en día, permite crear sistemas robustos, escalables y mantenibles gracias a diversas herramientas y métodos. Al igual que un arquitecto diseña edificios pensando en su funcionalidad y durabilidad, los arquitectos de software construyen la base de soluciones tecnológicas sólidas, adaptándose a la creciente complejidad del mundo digital y a la innovación constante en el campo.

-Aura María Fierro Fierro.

Artículo 3:

Implementación de Arquitectura de Software por el Dominio.

La implementación de DDD guía el desarrollo de sistemas eficientes, enfocándose en reglas del negocio y separando la complejidad técnica en capas. Integrando arquitecturas limpias y hexagonales, mejora pruebas, mantenimiento y adaptabilidad. Esto crea sistemas sostenibles, centrados en el negocio, con tecnología como soporte flexible.

-Camilo Andrés Bautista Cuellar.

Artículo 1:

Arquitectura de software: Fundamentos, teoría y práctica.

La arquitectura de software es clave para diseñar y mantener sistemas eficientes, guiando decisiones estructurales y asegurando propiedades no funcionales como rendimiento y escalabilidad. Es esencial en líneas de productos, reutilizando patrones arquitectónicos para

optimizar el desarrollo. En aplicaciones web, estilos como REST destacan por su escalabilidad e interoperabilidad, maximizando el valor y adaptabilidad del sistema.

-Carolina Martínez Cortes

Artículo 4:

Arquitectura para Herramienta de costos y programación.

El artículo presenta una herramienta educativa para la materia de “Costos, Presupuestos y Programación de obra” en ingeniería civil. Este software, basado en diagramas UML, ayuda a los estudiantes a gestionar APU, planificar actividades y optimizar recursos, fomentando el aprendizaje autónomo y simulando decisiones profesionales en proyectos de construcción.

-Cristian Fernando Narváez Sánchez

Artículo 24:

Documentación y análisis de los principales Frameworks de arquitectura de software en aplicaciones empresariales.

El artículo destaca Frameworks arquitectónicos clave para sistemas empresariales como ERP y CRM: en capas (mantenimiento y cambios aislados), cliente-servidor (roles distribuidos), y tres capas (modularidad y escalabilidad). Elegir el adecuado asegura sistemas escalables y fáciles de mantener.

-Cristian Jeanpool Bahamón Granados.

Artículo 1:

Análisis comparativo de patrones de diseño MVC y MVP para el rendimiento de aplicaciones web.

El artículo evalúa la eficiencia de los patrones de diseño MVC y MVP en el desarrollo web, destacando que MVC es más eficaz en términos de tiempo de desarrollo, líneas de código y uso de memoria. Esto proporciona una guía útil para elegir el patrón adecuado en proyectos futuros, optimizando el rendimiento.

-Dylan Santiago Narváez Pinto.

No realizo artículos.

-Isabela Córdoba Gutierrez.

No realizo artículos.

-Iván Andrés Murcia Epia

No realizo artículos.

-John Sebastián Penna Arias.

Artículo 12:

Arquitectura Hexagonal.

La Arquitectura Hexagonal, o "Ports and Adapters", aísla la lógica de negocio del sistema, permitiendo interacción con interfaces externas como bases de datos o servicios. Esto mejora la mantenibilidad y facilita las pruebas, aunque puede ser más compleja de implementar en aplicaciones simples.

-José Manuel Gasca Bonilla

Artículo 6:

Modelo y herramienta software para la gestión de riesgos en el desarrollo de aplicaciones web soportado en el estándar ISO/IEC 27005

Este artículo presenta un modelo de gestión de riesgos para aplicaciones web basado en la norma ISO/IEC 27005, con tres perspectivas: conceptual, lógica y física. Define fases como creación, evaluación e identificación de riesgos, utilizando diagramas UML y una estructura de base de datos. El modelo está diseñado para proyectos medianos y grandes, con ciclos de vida incrementales e iterativos. Refleja la importancia de integrar la gestión de riesgos desde el inicio y propone un enfoque sólido con controles e indicadores.

-Juan Pablo Betancourt Gómez

No se encontraba en clase.

-Julián David Fierro Casanova.

Artículo 1:

Una arquitectura para una herramienta de Patrones de Diseño

El artículo describe una arquitectura para integrar patrones de diseño en herramientas de desarrollo orientado a objetos, mejorando la eficiencia y reutilización del software. Utiliza patrones como Composite, Command y Observer para gestionar la interacción entre la interfaz y la lógica interna, ofreciendo vistas gráficas, jerárquicas y de código.

-Kevin Camilo Muñoz Campos.

No estaba en clase.

-Laura Valentina Ariza Alejo.

Artículo 11:

Mapeo de Arquitecturas de Software

El artículo analiza un mapeo sistemático sobre la recuperación de vistas arquitectónicas en sistemas software, destacando técnicas y mecanismos utilizados para representarlas. Resalta la importancia de la ingeniería inversa en el mantenimiento de software, especialmente en sistemas con poca o desactualizada documentación. Además, subraya la necesidad de marcos comunes para interpretar y reutilizar estas vistas de manera efectiva.

-Manuel Ricardo Diez Corredor.

No entrego artículos.

-Maria Del Mar Artunduaga Artunduaga.

Artículo 7:

Arquitectura de Microservicios para Desarrollo Web

El estudio analiza los desafíos de la arquitectura monolítica en la CGTIC de la Asamblea Nacional del Ecuador, como el mantenimiento y la escalabilidad. Propone adoptar microservicios, un enfoque modular y escalable, para mejorar el desarrollo de aplicaciones web, adaptándose mejor a los cambios y necesidades futuras.

-Maria José Murcia Martínez.

Artículo 1:

Patrones de Usabilidad 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 Prada.

Artículo 18:

Arquitectura de software para entornos móviles

La evolución de los sistemas operativos móviles hacia mayor estabilidad y robustez permite desarrollar aplicaciones más complejas y fiables. Este avance impulsa la creación de soluciones arquitectónicas móviles estandarizados que optimizan metodologías y procesos.

-Mariana Gonzáles Calderón

Artículo 2:

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

El artículo destaca la importancia de la arquitectura de software en proyectos tecnológicos, proponiendo un marco para seleccionar patrones como MVC, MVP y Microservicios según las necesidades del software. Este enfoque ayuda a optimizar calidad, rendimiento y mantenimiento, guiando a los desarrolladores en decisiones arquitectónicas informadas y sostenibles.

-Maydy Viviana Conde Ladino

Artículo 15:

Desarrollo de aplicaciones web utilizando el patrón de diseño Modelo/Vista/Controlador

El patrón MVC es esencial para sistemas interactivos, pero su aplicación en web es compleja por la partición temprana. La "Partición Flexible" permite implementar MVC de forma independiente a la arquitectura, evitando reestructurar el código al cambiar entre cliente/servidor.

-Mayra Alejandra Tamayo Perdomo.

Artículo 10:

Análisis comparativo de Patrones de Diseño de Software

El artículo destaca la importancia de los patrones de diseño para evitar duplicación de código y facilitar la reutilización. Analiza cinco patrones clave (MVC, MVP, MVVM, Template Method, Front Controller), evaluando sus ventajas, desventajas y aplicabilidad. Concluye que no hay un patrón superior, sino que cada uno se adapta a necesidades específicas, mejorando la organización y calidad del software. Este análisis ayuda a los desarrolladores a elegir el patrón adecuado según el contexto, promoviendo prácticas más eficientes y sostenibles.

-Patricia Sarmiento

Artículo 4:

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

El artículo aborda los desafíos de seguridad en arquitecturas de microservicios, proponiendo el patrón Microservice Security Pattern API Gateway (MSPAG) que usa JWT y H256 para centralizar la autenticación y proteger endpoints. Incluye un marco teórico, análisis del estado del arte e implementación en C#, Python y Java, evaluando su eficacia. Concluye que el uso de patrones como MSPAG mejora la seguridad, garantizando integridad y confidencialidad.

-Valentina Silva Garrido

Artículo 1:

Análisis comparativo de Patrones de Software.

Los patrones de diseño son fundamentales para crear software robusto y escalable. Cada uno ofrece soluciones específicas según el contexto y los requisitos del proyecto. Se comparan patrones como Template Method, MVC, MVP, Front Controller y MVVM, considerando factores como lenguaje, complejidad y seguridad. Al entender las ventajas y desventajas de cada patrón, los desarrolladores pueden tomar decisiones informadas para crear aplicaciones más eficientes y sostenibles. No existe un patrón perfecto; la elección depende de los requisitos del proyecto, el equipo y las características del software.

-Willian Steban González Cortes

Artículo 9:

Este artículo explora diferentes arquitecturas y metodologías de desarrollo para crear soluciones eficientes en sistemas de TI. Detalla la arquitectura de solución y de software, destacando estilos como capas, monolítica, microservicios, EDA y cliente-servidor, que organizan la comunicación y estructura de las aplicaciones. Subraya la importancia de elegir la arquitectura adecuada según las necesidades de la aplicación, como flexibilidad, independencia o escalabilidad. Además, destaca la metodología XP, que promueve la

comunicación continua entre desarrolladores y clientes, permitiendo cambios rápidos y manteniendo el enfoque en el cliente.

-Yordy Erik Nuñez Pineda

Artículo 7:

Introducción a los Patrones de Diseño

El artículo destaca la importancia de los patrones de diseño para abordar desafíos comunes en el desarrollo de software. Al seleccionar el patrón adecuado, se logran sistemas flexibles, escalables y con código de calidad. Además, enfatiza que los patrones no limitan la creatividad, sino que promueven una programación más segura y enfocada en la resolución de problemas. También se explica su clasificación en creacionales, estructurales y de comportamiento, con ejemplos prácticos de su aplicación y el uso de UML y POO, subrayando cómo mejoran la estructura, flexibilidad y durabilidad del software.

Exposición de los resúmenes y reflexión de Artículos de la ficha fin de semana.

-Jhon Alexander Corredor Medina

No se encuentra en clase.

-Yanuard Stevin Montialegre Bonilla

No se encuentra clase.

-Meyder Santiago Rodríguez

No se encuentra en clase

-Johan Calderón Perdomo

Artículo 1:

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

La arquitectura de software organiza sistemas definiendo sus elementos y relaciones, facilitando el desarrollo mediante la reutilización de componentes. Ha evolucionado a través de enfoques estructurales y patrones, lo que permite diseñar sistemas complejos de manera más eficiente y comprensible.

-Maryury Bonilla Gonzales.

Artículo 5:

Monolitos vs. Microservicios en Arquitectura de Software

Este estudio compara las arquitecturas monolíticas y de microservicios, analizando sus ventajas, desventajas y cuándo cada una es más eficiente. Se exploran casos de empresas como Amazon y eBay que migraron de una arquitectura a otra, destacando los beneficios y desafíos en escalabilidad, mantenimiento y desarrollo.

-Carlos Andrés Pantoja Jaramillo.

Artículo 8:

Impacto de implementaciones web del patrón MVC en los requisitos de calidad de percibidos.

El artículo analiza dos variantes de implementación del patrón Modelo-Vista-Controlador (MVC) en aplicaciones web y su impacto en atributos clave de calidad, como tiempo de respuesta y escalabilidad. Compara la versión clásica de MVC, basada en llamadas y respuestas, con una alternativa que utiliza tuberías y filtros implementados mediante cortinas en Python. La investigación concluye que, aunque la variante con tuberías muestra potencial, se necesitan más estudios para validar su aplicación práctica.

-Héctor Fabian Cardoso Morales

No está en clase.

-Stefany Nikoll Hidalgo

Artículo 16:

Desarrollo de una arquitectura de software para el robot Lázaro

La arquitectura, desarrollada en C#, se organiza en tres niveles: gestión de componentes básicos, librerías para aplicaciones de control, e interfaz de usuario con panel y simulador 3D. Su diseño modular mejora la reutilización y eficiencia, optimizando el desarrollo y control en robótica.

-Juan David Cerquera

Artículo 7:

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

Un estudio con desarrolladores identificó las arquitecturas más utilizadas:

- Arquitectura en la nube: Seguridad y flexibilidad (aplicaciones web).
- MVC: Mantenibilidad y modularidad (móviles, escritorio y web).

- MVP: Modificabilidad y rendimiento (móviles y web).
- Microservicios: Escalabilidad (aplicaciones web).

Cada arquitectura destaca según sus fortalezas y el tipo de dispositivo o aplicación objetivo.

-Erick Daniel Peña Cedeño

Artículo 2:

Arquitectura de Software basada en Microservicios para el Desarrollo Aplicaciones Web

Los microservicios representan una alternativa moderna y eficiente a las arquitecturas monolíticas, superando sus limitaciones en mantenimiento, escalabilidad y despliegue, al permitir que los componentes del sistema sean autónomos e independientes, facilitando así el desarrollo y mantenimiento de aplicaciones web.

-David Mauricio Flórez Quintero

No está en clase

-Jose Alejandro Osorio Ramírez

Problemas de internet.

-Laura Camila Sánchez

No está en clase

-Marlon Estiven Torres

Artículo 8:

Patrones de Diseño (XII): Patrones Estructurales – Flyweight

El artículo explora el patrón Flyweight, diseñado para optimizar el uso de memoria al compartir datos entre objetos similares. Se destacan ejemplos prácticos en sistemas donde la eficiencia es esencial, como aplicaciones gráficas y videojuegos. Además, se analizan sus ventajas, limitaciones y las claves para implementarlo correctamente, subrayando su importancia en escenarios de alto consumo de recursos.