ANYI:  
  
La arquitectura de software comenzó a ser muy conocido en los años 80 y desde entonces ha venido desenvolviéndose mas el concepto en los programadores al momento de realizar un proyecto, ha hecho eficaz el desarrollo y muy adaptable para entender cómo funciona este mismo y para realizar la codificación.

AURA:

El software guiado por el dominio ayuda a crear sistemas más sostenibles y eficientes. Ayuda la comunicación entre desarrolladores gracias al uso de un lenguaje en común.

CAMILO:

La arquitectura de software es la estructura o el boceto para llevar a cabo un proyecto con buenas bases, bases que sean adaptables y que se mantengan solidas a medida que se van implementando cambios.

CAROLINA:

La importancia de utilizar herramientas como Diagramas UML nos da la visibilidad para entender cómo funciona el sistema o el programa que se va a desarrollar, esto hace que las personas técnica y no técnicas tengan una idea de lo que se está haciendo.

JEANPOOL:  
  
se evaluó la eficiencia de los patrones de diseño en la realización de un proyecto, enfocándose en elegir el patrón más adecuado. Lo primero fue identificar y validar los patrones MVC y MVP existentes. Cada uno tuvo su propio proyecto para realizar la prueba, donde se evaluaron diferentes aspectos, como el tiempo de desarrollo, las líneas de código y el uso de memoria RAM, entre otros. Como resultado, se demostró que el MVC es mucho más eficaz en el desarrollo web.

PENNA:  
  
La arquitectura hexagonal permite al software interactuar con varias interfaces, permite la comunicacion con el mundo exterior.

GASCA:

**Resumen del modelo**

El modelo propuesto ofrece una estructura integral para gestionar los riesgos en el desarrollo de aplicaciones web. Se basa en el estándar ISO/IEC 27005 y propone una arquitectura con tres perspectivas:

* **Conceptual:** Define las fases del ciclo de vida del proyecto, desde la creación hasta la gestión de riesgos.
* **Lógica:** Utiliza diagramas UML para modelar las clases y relaciones del sistema.
* **Física:** Define la estructura de la base de datos.

JULIAN:

Ofrece una solución innovadora para la gestión de patrones de diseño en el desarrollo de software. Al proporcionar una arquitectura sólida y una interfaz de usuario intuitiva, esta herramienta puede mejorar significativamente la productividad y la calidad del software desarrollado.

ARIZA:

El estudio proporciona una visión general de la investigación en recuperación arquitectónica de software. Los resultados obtenidos son valiosos para investigadores, desarrolladores y profesionales de la ingeniería de software que buscan comprender y mejorar la gestión de la evolución de los sistemas.

MAR:

El proceso de desarrollo de software de la CGTIC de la Asamblea Nacional del Ecuador emplea una arquitectura monolítica, lo que ha generado dificultades en mantenimiento, escalabilidad y entregas. Este estudio tuvo como objetivo identificar las tecnologías, metodologías y arquitecturas utilizadas, y explorar alternativas para la implementación de microservicios. Con un enfoque cualitativo, se aplicó un grupo focal y revisión bibliográfica sobre microservicios. El análisis identificó el estado del arte y los requisitos de software para desarrollar aplicaciones web, proponiendo una arquitectura de software adecuada.

MAJO:

El proyecto STATUS propone integrar la usabilidad en el software desde la fase de diseño, utilizando patrones como "deshacer", "cancelar", "múltiples idiomas" para mejorar la interacción del usuario. A diferencia de los enfoques tradicionales, que evalúan la usabilidad solo al final, STATUS utiliza un método inductivo que permite ajustar el diseño de forma continua durante el proceso. Esta integración estructural garantiza que el sistema sea intuitivo y accesible desde el principio, sin grandes pérdidas de tiempo o recursos, mejorando la experiencia del usuario en cada fase del desarrollo.

MARIANA CHARRY:

La evaluación de los sistemas operativos está causando un gran impacto, porque cada vez son más estables y robustos, lo que permite a los desarrolladores de software, crear aplicaciones móviles de mayor tamaño y complejidad. 1 El objetivo del artículo fue definir una solución arquitectónica móvil que compartiera algunos de los principios más reconocidos de la arquitectura de software en general, y de 2 esa forma, ayudar a estandarizar y adaptar metodologías, métodos, procesos y enfoques.

MARIANA G:

Presenta un marco de trabajo para seleccionar patrones arquitectónicos que mejoren la calidad, rendimiento, mantenimiento y adaptabilidad del software. Este marco guía a desarrolladores y arquitectos para tomar decisiones basadas en características del proyecto (como tipo de desarrollo y requisitos clave). Se describen patrones como MVC, MVP, Microservicios y Arquitectura en la Nube, y su idoneidad según el contexto y necesidades del software. Finalmente, se valida el marco mediante un caso práctico en el que el usuario selecciona opciones para obtener recomendaciones arquitectónicas personalizadas.

MAYRA:

Aborda la importancia de los patrones de diseño como soluciones estandarizadas a problemas comunes en el desarrollo de software, destacando su capacidad para evitar la duplicación de código y facilitar la reutilización. Se analizan cinco patrones clave: Model-View-Controller (MVC), Model-View-Presenter (MVP), Model Front Controller, Template Method y Model-View-ViewModel (MVVM), evaluando sus componentes, ventajas y desventajas, así como su aplicabilidad en diferentes contextos. La investigación revela que no existe un patrón superior, dado que cada uno cumple un propósito específico, y concluye que los patrones de diseño son fundamentales para mejorar la organización, el mantenimiento y la calidad del código en sistemas de software, al permitir una estructura coherente que facilita su comprensión y evolución en el tiempo.

MAIDY:

El modelo de diseño mvc es útil para diseño de software interactivos se encarga de manejar datos y la lógica de la aplicación, El patrón MVC es muy útil para organizar sistemas interactivos y aplicaciones web. Sin embargo, dividir las partes de una aplicación demasiado pronto puede complicar las cosas. La propuesta de "Partición Flexible" permite aplicar MVC sin cambiar mucho el código cuando hay modificaciones en la arquitectura, haciendo el desarrollo más fácil y adaptable.

PATRICIA:

El artículo presenta un marco de trabajo para seleccionar patrones arquitectónicos en el desarrollo de software, abordando problemas de desacoplamiento y falta de conocimiento arquitectónico que afectan la calidad del producto. Basado en cinco fases, identifica patrones clave (como MVC, Microservicios y Arquitectura en la Nube) y establece reglas para elegirlos según el tipo de desarrollo y las características requeridas. Validado mediante un prototipo, el marco mejora tiempos, calidad, escalabilidad y mantenibilidad. Concluye que la solución es una herramienta práctica para guiar a arquitectos y desarrolladores en la selección de patrones adecuados, estableciendo estructuras desde el inicio del proyecto.

VALENTINA SILVA:

Los patrones de diseño son herramientas esenciales para construir software robusto y escalable. Cada patrón ofrece una estructura y solución específicas, pero la elección correcta depende del contexto y los requisitos del proyecto.

En este análisis, comparamos patrones populares como Template Method, MVC, MVP, Front Controller y MVVM, considerando aspectos como complejidad de programación, legibilidad, seguridad y uso.

WILLIAM:

Este artículo de reflexión explora diferentes arquitecturas de software y metodologías de desarrollo utilizadas para crear soluciones eficientes y flexibles en sistemas de TI. Primero, detalla la arquitectura de solución, que guía el diseño de estructuras integradas para responder a necesidades actuales y futuras, y la arquitectura de software, que tiene patrones y tecnologías específicas. Luego, describe estilos arquitectónicos como capas, monolítico, microservicios, EDA (Arquitectura Orientada a eventos) y cliente-servidor, que organizan la comunicación y estructura de aplicaciones.

YORDY:

Habla sobre los patrones de diseño, que son como "recetas" para resolver problemas comunes al programar. Estos patrones ayudan a crear programas más organizados, fáciles de entender y más eficientes.

El artículo explica por qué es importante usar estos patrones y cómo elegir el adecuado para cada situación.

JOHAN CALDERON:

La Arquitectura de Software organiza un sistema a nivel alto, definiendo elementos y sus relaciones. Facilita el desarrollo al reutilizar componentes mediante arquitecturas de referencia. Ha evolucionado a través de enfoques estructurales, patrones y escenarios, guiando el diseño y la comprensión de sistemas complejos.

MARYURY:

Compara la arquitectura monolítica y de microservicios al momento de construir el software. El articulo se refiere a las empresas que han adoptado proyectos monolíticos, actualmente buscan modularizar la arquitectura que poseen mediante microservicios. Las ventajas de los microservicios es la escalabilidad y los microservicios.

CARLOS ANDRÉS:

Se ha estudiado el patrón de diseño Modelo-Vista-Controlador (MVC) en aplicaciones web, específicamente dos formas diferentes de implementarlo. Se analizó cómo estas diferentes implementaciones afectan la calidad percibida por los usuarios, como la rapidez con la que responde la aplicación y su capacidad para manejar muchas solicitudes al mismo tiempo.

NIKOLL:

se centra en el desarrollo de una arquitectura de software específica para el robot móvil Lázaro. Esta arquitectura se divide en tres niveles principales:

1. **Primer Nivel:** Encargado de gestionar los componentes básicos del robot. Esto incluye los elementos fundamentales que permiten al robot funcionar, como sus sensores, actuadores y la electrónica de bajo nivel.
2. **Segundo Nivel:** Proporciona una serie de librerías que facilitan la creación de aplicaciones de control para el robot. Estas librerías ofrecen una capa de abstracción sobre los componentes básicos, permitiendo a los desarrolladores centrarse en la lógica de control sin tener que preocuparse por los detalles de bajo nivel.
3. **Tercer Nivel:** Ofrece una interfaz de usuario amigable, con un panel de control y un simulador 3D. Esta interfaz permite a los usuarios interactuar con el robot de manera intuitiva, programar sus movimientos y monitorear su estado en tiempo real.

JUAN CERQUERA:

Se realizó un estudio a desarrolladores y arquitectos de software profesionales sobre cuáles son las distintas arquitecturas de software más usadas. Luego de analizar las distintas arquitecturas, se define cuáles son las más usadas y en qué se destaca cada una.

Siendo las más usadas:

* **Arquitectura en la nube:** Seguridad y flexibilidad.
* **MVC:** Mantenibilidad, rendimiento, velocidad y memoria.
* **Microservicios:** Mantenibilidad, rendimiento, seguridad y flexibilidad.
* **MVP:** Modificabilidad, rendimiento, estabilidad, flexibilidad, modularidad.

También se analizaron los distintos dispositivos a los que estaban enfocadas esas arquitecturas:

* **Arquitectura en la nube:** Aplicaciones web
* **MVC:** Dispositivos móviles, aplicaciones de escritorio, aplicaciones basadas en la web
* **Microservicios:** Aplicaciones web
* **MVP:** Dispositivos móviles y aplicaciones web

ERICK:

El documento sobre Microservicios explora el desarrollo de una arquitectura de software más flexible y moderna que busca superar las limitaciones de la tradicional arquitectura monolítica. La investigación se enfoca en cómo los microservicios ofrecen una solución más eficiente y adaptable para el desarrollo y mantenimiento de aplicaciones, al permitir que los componentes del sistema funcionen de manera autónoma e independiente.

MARLÓN:

Este artículo explica una técnica en programación llamada "patrón Flyweight". Esta técnica sirve para ahorrar espacio en la computadora cuando se trabaja con muchos objetos que son muy parecidos entre sí. En vez de crear cada objeto desde cero, se comparte la información que tienen en común, lo que hace que se use menos memoria.