Fundamentos de arquitectura de software:
Patrones y estilos arquitectónicos

Resultado de Aprendizaje (RA1): Comprende los fundamentos de la arquitectura de software, incluyendo conceptos clave, patrones arquitectónicos y principios de diseño, en el contexto de proyectos de desarrollo de software empresarial.

Resultado de Aprendizaje (RA2): Aplica efectivamente los principios de diseño arquitectónico y las metodologías ágiles en proyectos de desarrollo de software, materializando soluciones de alta calidad que optimizan procesos empresariales y mejoran la productividad de la compañía.

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA
"Diseño y prestación de servicios de docencia, investigación y extensión de programas de pregrado, aplicando todos los requisitos de las normas ISO implementadas en sus sedes Neiva y Pitalito"

Cronograma

Encuentro sincrónico	Fecha
1	Martes, 10 de junio de 2025, **Hora**: 7:00 p.m. a 8:00 p.m.
2	Jueves, 12 de junio de 2025, **Hora**: 7:00 p.m. a 8:00 p.m.
3	Martes, 17 de junio de 2025, **Hora**: 7:00 p.m. a 8:00 p.m.
4	Jueves, 19 de junio de 2025, **Hora**: 7:00 p.m. a 8:00 p.m.
5	Martes, 24 de junio de 2025, **Hora**: 7:00 p.m. a 8:00 p.m.
6	Jueves, 26 de junio de 2025, **Hora**: 7:00 p.m. a 8:00 p.m.
7	Martes, 1 de julio de 2025, **Hora**: 7:00 p.m. a 8:00 p.m.
8	Jueves, 3 de julio de 2025, **Hora**: 7:00 p.m. a 8:00 p.m.

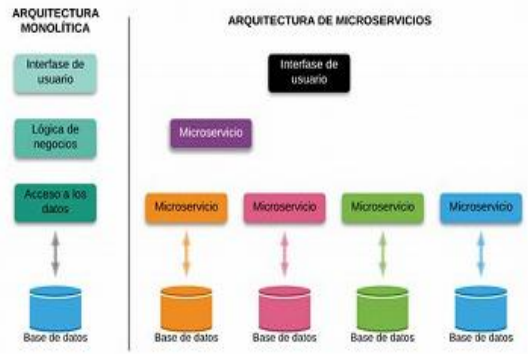
Introducción

Visión general del curso

Este curso aborda los fundamentos de la arquitectura de software, explorando conceptos clave, patrones y estilos arquitectónicos para construir sistemas robustos y escalables.

Relevancia de la arquitectura en sistemas eficientes

La arquitectura de software define la estructura y la interacción de componentes, asegurando que los sistemas cumplan requisitos funcionales y no funcionales como escalabilidad y mantenibilidad.



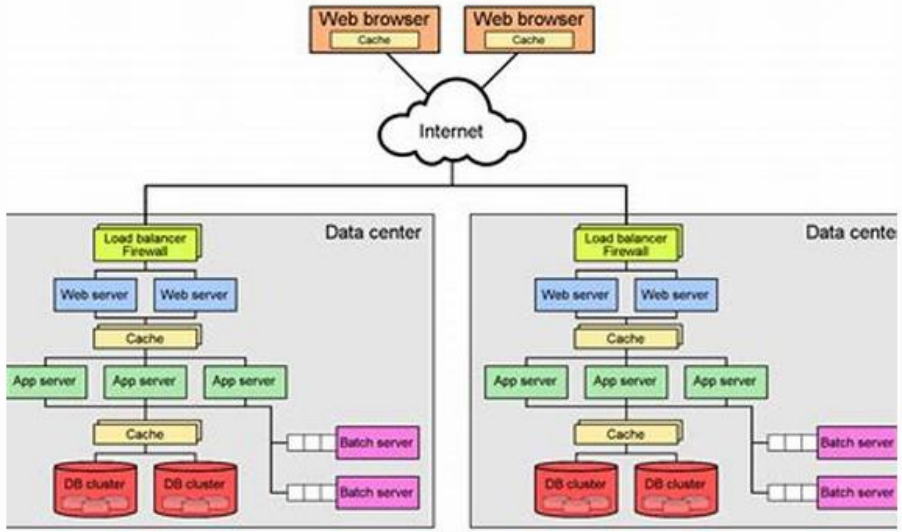
Contexto tecnológico actual

Ante la rápida evolución tecnológica y la adopción de tecnologías modernas como la nube y microservicios, la arquitectura de software es esencial para diseñar soluciones flexibles y sostenibles.

Introducción a la arquitectura de software

Definición de arquitectura de software

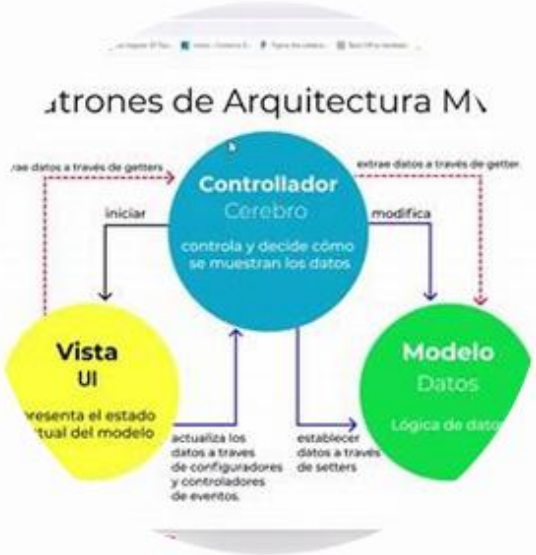
La arquitectura de software es la estructura organizativa y conceptual que guía el diseño y desarrollo de sistemas, asegurando que se cumplan requisitos funcionales y no funcionales como escalabilidad, rendimiento y seguridad.



Importancia en la estructuración y mantenimiento de sistemas

Una buena arquitectura facilita la modularidad, reutilización y escalabilidad, permitiendo una evolución eficiente del software y una comunicación clara entre equipos, lo que optimiza costos y mantiene la flexibilidad del sistema.

Patrones de diseño arquitectónico

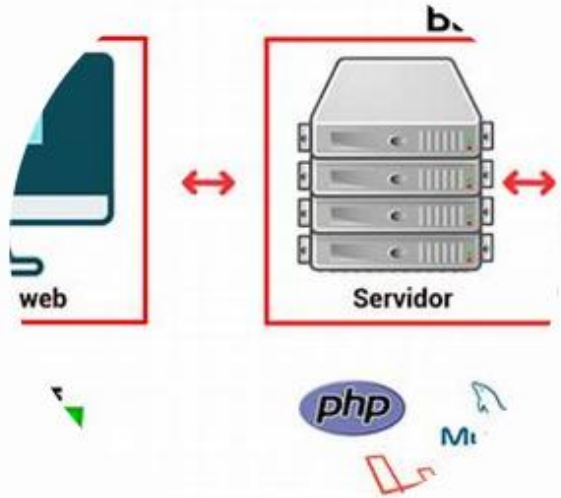


MVC

Separación de responsabilidades: El patrón Modelo-Vista-Controlador (MVC) divide la aplicación en tres componentes: Modelo, Vista y Controlador, facilitando la modularidad y mantenibilidad mediante la separación clara de datos, interfaz y lógica de control.

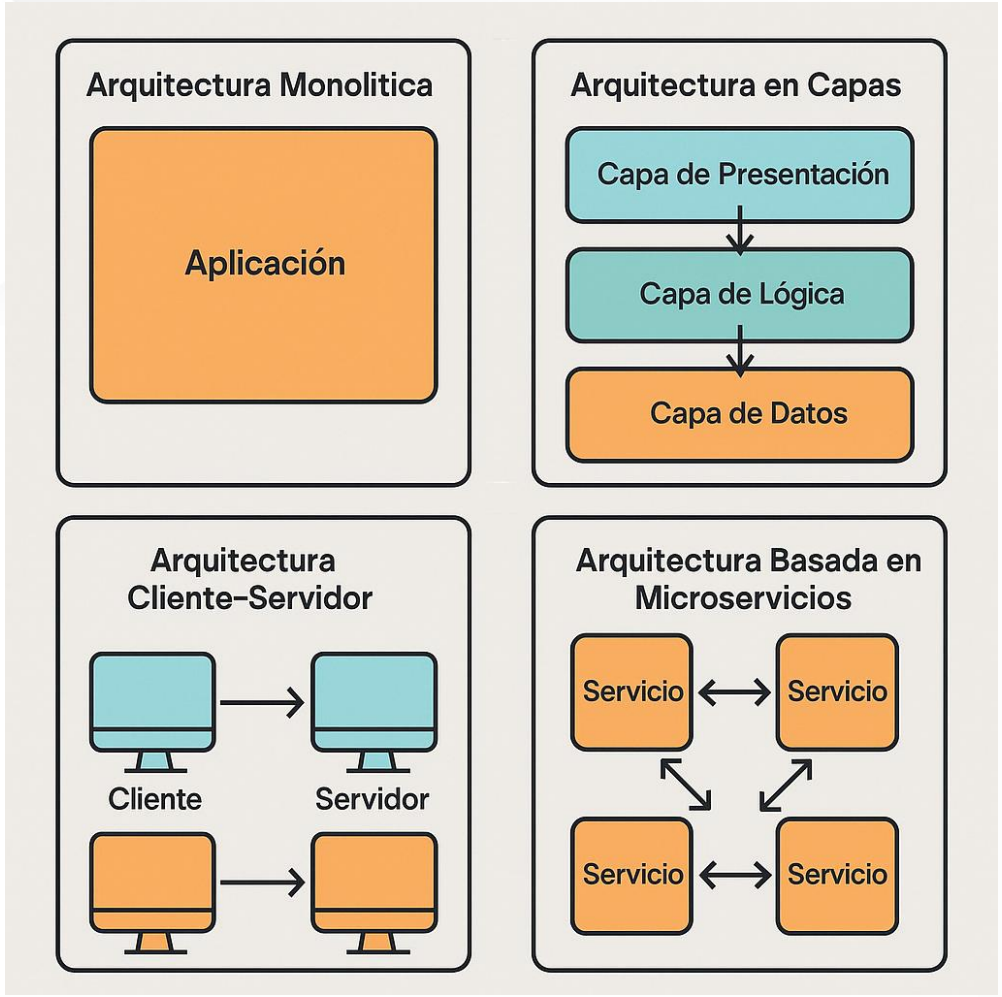
Arquitectura en capas

Organización jerárquica: La arquitectura en capas organiza el sistema en niveles con responsabilidades definidas, como la capa de presentación para la interfaz y otras capas para lógica y datos, mejorando la escalabilidad y testabilidad.



Separación Frontend-Backend

Esta división establece una clara separación entre la interfaz de usuario (frontend) y la lógica de negocio y datos (backend), permitiendo desarrollos independientes y mejor mantenimiento del sistema.



Arquitecturas monolíticas vs. distribuidas

1 Definición de arquitecturas monolítica y distribuida

La arquitectura monolítica integra todos los componentes de una aplicación en una única unidad indivisible, mientras que la arquitectura distribuida fragmenta la aplicación en servicios pequeños e independientes que se comunican mediante protocolos ligeros.

2 Ventajas y desventajas de arquitecturas monolíticas y distribuidas

Las arquitecturas monolíticas ofrecen simplicidad en desarrollo y mejor rendimiento interno, pero presentan limitaciones en escalabilidad y mantenimiento. Las distribuidas permiten escalabilidad y flexibilidad tecnológica, aunque aumentan la complejidad en gestión y comunicación entre servicios.

Estilos arquitectónicos modernos

1

Arquitectura Hexagonal

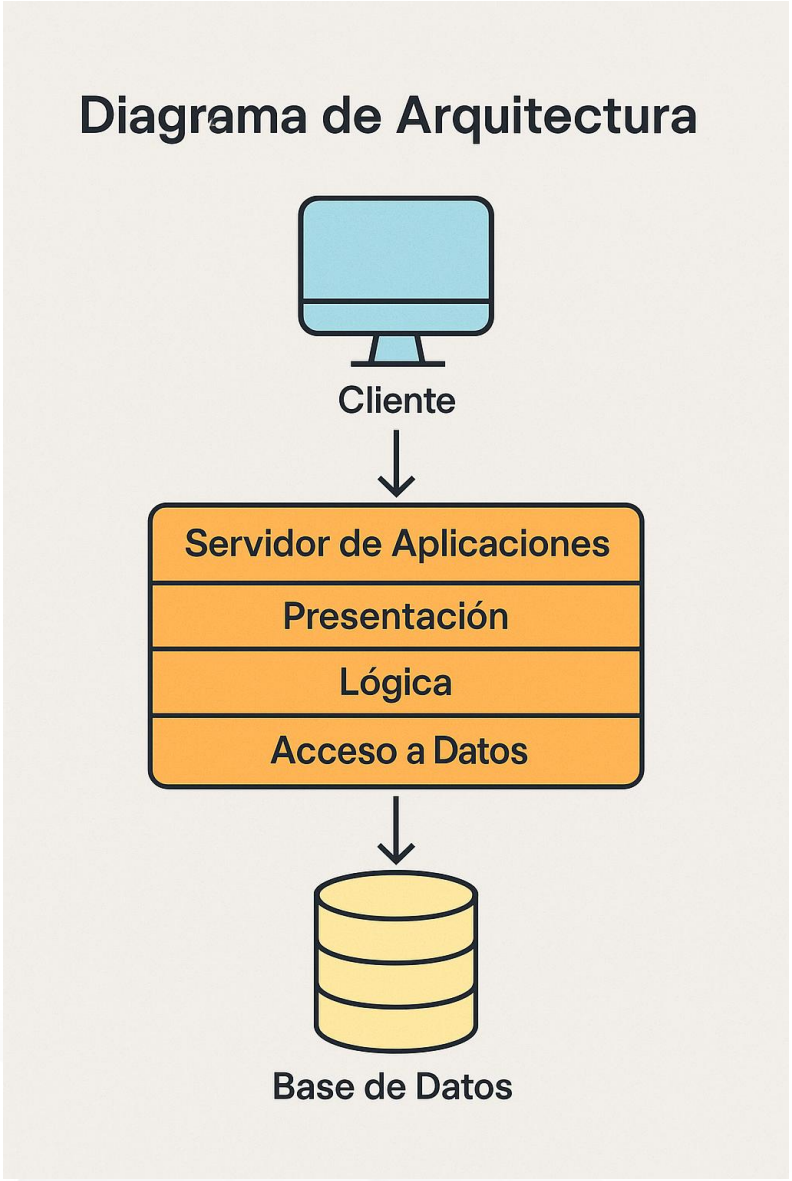
Aaislamiento del núcleo de negocio: La Arquitectura Hexagonal protege el núcleo de negocio mediante puertos y adaptadores, facilitando el desacoplamiento y la independencia tecnológica.

2

Arquitectura Onion

Ccapas concéntricas para separación clara: La Arquitectura Onion organiza la aplicación en capas concéntricas, asegurando que el dominio central sea independiente y separado de las infraestructuras externas.

Comparativa visual de estilos arquitectónicos



Comparación de estilos arquitectónicos

Se presenta una tabla comparativa que destaca las características clave de MVC, capas, Hexagonal, Onion y CQRS, facilitando la comprensión de sus diferencias y similitudes en estructura, responsabilidades y complejidad.

Representación visual y diseño gráfico

El gráfico esquemático utiliza líneas finas y tonos violetas sobre fondo oscuro para mostrar la posición relativa y relaciones entre los estilos, enfatizando la simplicidad de MVC y capas, el aislamiento en Hexagonal y Onion, y la segregación funcional en CQRS.

Casos de uso reales

1

Aplicación de Arquitectura Monolítica

La arquitectura monolítica es adecuada para proyectos pequeños o startups que requieren desarrollo rápido y bajo costo, facilitando la gestión con equipos reducidos.

2

Migración a Microservicios en Netflix

Netflix migró de un monolito a microservicios en la nube para mejorar la escalabilidad y resiliencia, permitiendo miles de despliegues diarios y mayor agilidad.

3

Uso del Patrón MVC en Aplicaciones

El patrón Modelo-Vista-Controlador facilita la separación de responsabilidades en aplicaciones web y móviles, mejorando la mantenibilidad y permitiendo trabajo simultáneo en componentes.

4

Arquitecturas Hexagonal y Onion para Flexibilidad

Estas arquitecturas promueven el desacoplamiento y la independencia tecnológica, facilitando pruebas unitarias y la evolución del sistema con alta flexibilidad.

Buenas prácticas en selección de patrones y estilos

Selección de arquitecturas según el tipo de proyecto

Es fundamental evaluar la complejidad y tamaño del proyecto para elegir patrones adecuados, desde MVC para proyectos pequeños hasta microservicios para sistemas escalables y modulares.

Consideraciones sobre escalabilidad y mantenimiento

Optar por arquitecturas que permitan escalar componentes independientemente y que promuevan bajo acoplamiento facilita el mantenimiento y mejora el rendimiento del sistema.

Importancia del equipo y la documentación

La arquitectura debe adaptarse al tamaño y experiencia del equipo, mientras que la documentación y pruebas aseguran la calidad y evolución controlada del software.