“DISEÑO ARQUITECTÓNICO DE INGENIERÍA DE SOFTWARE”

INGENIRIA DE SOFTWARE

INGENIERIA EN SISTEMAS COMPUTACIONALES

PRESENTA:

EDGAR CORTÉS RESÉNDIZ

LUIS ANTONIO DIAZ ROSALES

JIQUILPAN, MICHOACÁN, MARZO DE 2025

Actividad 4: Diseño arquitectónico de ingeniería de software

Es una fase calve del desarrollo de sistemas, en donde se llegan a definir la estructura base de un sistema o aplicación, con el objetivo de poder establecer una organización en los componentes en la interacción que existe entre ellos y sus principales principios que guían su desarrollo.

Sus principales características consisten en establecer una estructura del sistema en donde se definen sus componentes y su interconexión, existe también los patrones de diseño, su escalabilidad y rendimiento, además de que sus tecnologías y sus plataformas. Con esto podemos obtener una fase crucial en el desarrollo del sistema, esto asegurando que exista una solución eficiente y mantenible para poder entregar al cliente.

El diseño arquitectónico en ingeniería de software es como planificar una ciudad: necesitas una estructura bien pensada para que todo funcione de manera eficiente y pueda crecer con el tiempo. Imagina que estás construyendo un sistema, como una plataforma de gestión escolar. Para organizarlo, divides el software en módulos o componentes, cada uno con una función específica. Por ejemplo, un módulo para gestionar usuarios (estudiantes, profesores y administrativos), otro para las calificaciones y la asistencia, uno más para los pagos de colegiaturas y otro para generar informes. Aunque cada módulo tiene su propia tarea, todos trabajan juntos para ofrecer una solución completa.

Para que todo funcione de manera ordenada, se utilizan patrones de diseño, que son como "recetas" probadas para resolver problemas comunes. Uno de los más conocidos es el Modelo-Vista-Controlador (MVC), que separa el sistema en tres partes: el Modelo (que maneja la lógica del negocio y los datos), la Vista (que se encarga de la interfaz gráfica) y el Controlador (que gestiona la interacción del usuario). Por ejemplo, en un sitio de comercio electrónico, el Modelo maneja los productos, la Vista los muestra en la pantalla y el Controlador procesa las compras. Otro patrón popular es el de microservicios, donde cada funcionalidad del sistema es un servicio independiente, como en Netflix, que tiene servicios separados para gestionar cuentas, recomendaciones, reproducción de vídeos y facturación.

Pero no basta con que el sistema funcione hoy; también debe estar preparado para crecer mañana. Aquí entra la escalabilidad, que es la capacidad del software para adaptarse a nuevas demandas sin perder rendimiento. Hay dos formas de escalar: verticalmente (mejorando el hardware, como añadir más RAM o un procesador más potente) y horizontalmente (añadiendo más servidores para distribuir la carga). Por ejemplo, una aplicación de banca en línea podría añadir nuevas funciones, como pagos con QR, sin necesidad de modificar toda la estructura del sistema.

La seguridad y el rendimiento también son pilares fundamentales. Un buen diseño arquitectónico debe proteger los datos y optimizar el uso de los recursos. Para la seguridad, se usan técnicas como la autenticación y autorización (por ejemplo, con tokens JWT), la encriptación de datos sensibles (usando algoritmos como AES o RSA) y la protección contra ataques con firewalls y sistemas de detección de intrusos. En cuanto al rendimiento, se pueden implementar estrategias como el uso de caché, que almacena temporalmente datos para reducir la carga en los servidores, algo muy útil en sitios con millones de visitas diarias, como un portal de noticias.

Además, es importante elegir las tecnologías adecuadas para cada parte del sistema. Por ejemplo, en una aplicación móvil, podrías usar React Native o Flutter para el frontend (la parte que ven los usuarios), Node.js o Python con Django para el backend (la lógica del negocio), PostgreSQL o Firebase para la base de datos, y servicios en la nube como AWS o Google Cloud para el almacenamiento y procesamiento de datos.

Una de las formas más comunes de organizar un sistema es mediante la arquitectura en capas, que divide el software en niveles, cada uno con una función específica. Por ejemplo, en un sistema de streaming como Netflix, la capa de presentación es la interfaz que ven los usuarios (la app o la página web), la capa de lógica de negocio se encarga de verificar si el usuario tiene una suscripción activa o si el contenido está disponible en su país, la capa de acceso a datos recupera la información de los vídeos y usuarios desde la base de datos, y la capa de base de datos almacena todo, desde los detalles de los vídeos hasta el historial de reproducción.

Esta arquitectura tiene muchas ventajas: facilita el mantenimiento, permite reutilizar código, mejora la seguridad al restringir el acceso a ciertas capas y hace que el sistema sea más escalable. Sin embargo, también tiene sus desafíos, como una mayor complejidad y la posibilidad de que el acceso a datos sea más lento debido a la separación de capas. Además, si las capas están demasiado acopladas, puede ser difícil hacer cambios rápidos.

En resumen, un buen diseño arquitectónico es la base para construir software robusto, seguro y adaptable. Es como tener un plano bien pensado antes de construir una casa: si la estructura es sólida, todo lo demás fluirá de manera natural, y podrás añadir nuevas habitaciones (o funcionalidades) sin que todo se derrumbe.