**ARQUITECTURA DE SOFTWARE:**

**Sistemas de Gestión de Datos:**

Administran un gran volumen de datos y el dominio de la aplicación se centra en el manejo de sus datos. Poseen un gran dinamismo en sus datos: muchas altas, bajas y modificaciones a diario.

Usuarios intensivos: son usuarios que utilizan el sistema en forma constante, todos los días, varias veces al día.

Usuarios casuales: son usuarios que acceden al sistema en forma esporádica.

|  |  |  |  |
| --- | --- | --- | --- |
| **Arquitectura** | **Características** | **Ventajas** | **Desventajas** |
| Monocapa. | \*Monousuario.  \*Denominados “desktop”.  \*No posee conectividad. No interactúa con otra aplicación, salvo con el SO en el cual fue instalado.  \*Persistencia: los datos se guardan en archivos propios de la aplicación. | \*Simple de construir.  \*Facilidad de encontrar bugs.  \*Acceso directo a datos.  \*Fácil puesta en marcha. | \*Integración de Datos. Debe realizarse una consolidación mano a mano o realizar procesos batch que corran fuera de hora para integrar los datos.  \*Se encuentran acopladas al SO para el cual fueron desarrolladas. |
| Cliente-Servidor | Aplicación Cliente: componente Desktop, instalado en la PC de cada usuario.  Aplicación Servidor: servidor centralizado, con una Base de Datos que hace de repositorio de la aplicación. | \*Al incorporar una BdD se obtienen ventajas como: concurrencia, carga de datos simultáneas, integridad de datos.  \*Se eliminan los problemas de integración de datos.  \* Se delega gran parte de la complejidad del sistema en la base de datos.  \* Los usuarios pueden estar en diferentes lugares físicos, y acceder al mismo servidor de base de datos. | \*Algunos problemas de conectividad o acceso.  \* Instalar cada aplicación cliente en la PC.  \* Aumenta la complejidad de actualización.  \*Aumenta la complejidad del rastreo de bugs.  \*Problemas de seguridad.  \*Al igual que la arquitectura en Monocapa, la aplicación cliente normalmente se encuentra acoplada al sistema operativo. |
| Multicapa | | | |
| Multicapa con Cliente Desktop | \*El cliente sigue utilizando una aplicación instalable, pero no accede directamente a los datos, sino que lo hace a través de la aplicación servidor.  \*El Cliente se considera liviano, porque no posee la aplicación entera, sino sólo una cáscara del sistema. Muchas reglas del negocio se implementan en la aplicación servidor.  \*Uno de los temas centrales que surgen en esta arquitectura, es el método de comunicación que se va a utilizar entre la aplicación cliente y la aplicación servidor. \* La aplicación cliente ahora no puede aprovechar los beneficios de un driver de base de datos, sino que debe utilizar algún método propio de conexión.  \*Existen diversas tecnologías para solucionar este problema, dependiendo del lenguaje utilizado: – Comunicación tradicional a bajo nivel, mediante la utilización de Sockets  – Comunicación mediante RPC (Remote Procedure Call) o RMI (Remote Method Invocation): ambas alternativas presentes en la mayoría de los lenguajes de programación actuales, proveen mecanismos automáticos para la ejecución de código remoto, junto con la serialización de los parámetros de entrada y salida. – Mediante el uso de WebServices. La aplicación cliente puede utilizar el protocolo HTTP para intercambiar mensajes con la aplicación servidor, mediante archivos en formato XML. | \*Interfaz gráfica más robusta de lo que se puede realizar con una tecnología web.  \*Se eliminan aspectos de seguridad ya que las validaciones de negocios pueden ser programadas por el servidor de aplicaciones.  \*Los programas instalados en las PC suelen ser más livianos, por lo que la actualización es más fácil. | \*Si bien es más fácil de actualizar que en el cliente-servidor sigue siendo necesario el módulo de actualización.  \*Lógica divida en 3 partes, incrementa la dificultad de encontrar bugs.  \* La interacción de la capa Servidor en los pedidos a la base de datos puede representar un cuello de botella en la performance de la aplicación.  \*Si no se utiliza WebServices para la comunicación entre la aplicación cliente y la aplicación servidor, esta arquitectura presenta los mismos problemas de conectividad que la arquitectura Cliente-Servidor.  \*Acoplamiento del cliente al sistema operativo. |
| Multicapa con Cliente Web | \* Desarrollo de aplicaciones que se ejecuten dentro de un  Browser de Internet.  \*No son simples aplicaciones o portales sino aplicaciones enteras y complejas, con mucha lógica de negocio involucrada.  \* Esta arquitectura es una de las más recientes y se encuentra en constante evolución.  \*Composición:  -Aplicación cliente ejecutando en un Browser de Internet.  -Application Server. Al tratarse de aplicaciones Web, el servidor debe poder lidiar con el protocolo HTTP  -Base de datos | \*Elimina problemas de acceso presente en las demás arquitecturas (Puerto 80 y protocolo HTTP permitidos).  \*Elimina problemas de instalación.  \*No es necesario actualizarlo, el usuario siempre accede a la última versión.  \*Se elimina problemas de acoplamiento al SO.  \*Ágiles para usuarios comunes.  \*El usuario solo posee una capa de la solución, lo que le da mayor seguridad. | \*Es necesario tener el browser perfectamente configurado.  \* La performance del protocolo HTTP deja mucho que desear a la hora transferir grandes  volúmenes de datos.  \*Calidad de interfaz gráfica menor a la que se logra en un Cliente Desktop.  \*Dependientes del Browser.  \*Sin acceso a internet, el usuario no puede ingresar a la aplicación.  \* Mayor tiempo de desarrollo que en un Cliente Desktop. |
| Arquitectura Orientada a Servicios (SOA): | Aquella que soporta un enfoque basado en un diseño de servicios que reflejan las actividades de negocio del mundo real de acuerdo con los procesos de negocio de la organización.  7 capas entre proveedores y consumidores de servicios  El problema es lograr una arquitectura en la cual los procesos de negocio son soportados por sistemas que permiten el libre intercambio de información. Cada proceso de negocio tiene su propio sistema.  Con SOA, las aplicaciones se reemplazan por servicios que interactúan entre sí. | \*Mejora de performance de procesos del negocio  \*Reducción de costos de IT  \* Mejora de la efectividad de las operaciones del negocio  \* Mejora de la efectividad de IT  \* Mejora de la eficacia de gestión  \* Reducción de riesgos |  |