UDE
UNIVERSIDAD DE LA EMPRESA

Bases de Datos 3

Docentes: Federico Gómez, Diego Siri

Carreras: Licenciatura & Ingeniería en Informática

Año de la carrera: 3º

UDE
UNIVERSIDAD DE LA EMPRESA

Capítulo 1: Arquitecturas

UDE
UNIVERSIDAD DE LA EMPRESA

Introducción:

Toda aplicación que trabaja con bases de datos lo hace con el objetivo de ***persistir*** en ellas la información manejada por la aplicación. El concepto de ***persistencia*** significa que los datos permanezcan almacenados luego de finalizada la ejecución.

Las ***bases de datos*** representan el mecanismo más confiable y comúnmente utilizado para persistir los datos de una aplicación, aunque también existen otros mecanismos (archivos de texto plano, archivos binarios, archivos XML, etc.).

Para que el manejo de la persistencia (independientemente de cuál sea el mecanismo concreto utilizado) se realice de forma organizada y eficiente, es fundamental estudiar previamente el concepto de ***arquitectura*** de una aplicación.

UNIVERSIDAD DE LA EMPRESA

Arquitectura de una aplicación:

No existe una definición universal del concepto de ***arquitectura*** de una aplicación. En general se suele referir a la manera en que los distintos ***componentes*** de la aplicación se organizan y comunican entre sí. Esta es la acepción que adoptaremos en este curso.

Así mismo, tampoco existe una definición universal del concepto de ***componente***. Usualmente se refiere a un "pequeño" subsistema, (conjunto de módulos, clases, etc.) que reside en un equipo o dispositivo y que tiene una utilidad específica y claramente definida.

Cuando se habla de ***arquitectura*** de una aplicación, normalmente se habla de dos tipos de arquitectura:

- Arquitectura ***física*** de la aplicación.
- Arquitectura ***lógica*** de la aplicación.

UNIVERSIDAD DE LA EMPRESA

Arquitectura física de una aplicación:

La arquitectura ***física*** de una aplicación se refiere a la manera en la que los distintos componentes de la misma se ***distribuyen*** entre los distintos equipos sobre las cuales corre la aplicación.

En este contexto, se dice que la aplicación posee una arquitectura ***standalone*** cuando todos sus componentes residen en el mismo equipo. En este tipo de aplicaciones, todos sus componentes se ejecutan localmente, sin necesitar interactuar con otros equipos.

Por ejemplo, cualquier aplicación de escritorio posee una arquitectura standalone, como así también cualquier software de oficina (procesador de texto, planilla de cálculo, presentador de diapositivas, etc.).

UNIVERSIDAD DE LA EMPRESA

Arquitectura física de una aplicación (continuación):

Se dice que la aplicación posee una arquitectura ***distribuida*** cuando sus componentes se encuentran repartidos entre distintos equipos y se comunican entre sí a través de la red. Aplicaciones con este tipo de arquitectura son cada vez más utilizadas, poseen mayor tamaño (cantidad de componentes) y más complejidad.

Tradicionalmente, las arquitecturas distribuidas se suelen clasificar a su vez en dos tipos (aunque también puede ser en ***más*** tipos):

- Arquitectura ***Cliente-Servidor***.
- Arquitectura ***N-tier***.

Cliente-Servidor es la más simple de las arquitecturas distribuidas, y fue el primer tipo de arquitectura distribuida en surgir. ***N-tier*** es un tipo de arquitectura distribuida que surgió posteriormente, como una evolución a la arquitectura cliente-servidor.

UNIVERSIDAD DE LA EMPRESA

Arquitectura Cliente-Servidor:

Una arquitectura Cliente-Servidor consiste en:

- Un ***único*** equipo denominado ***Servidor***. Brinda servicios a los Clientes residentes en otros equipos.
- ***Uno o varios*** equipos denominados ***Clientes***, las cuales se comunican remotamente con el Servidor para usar sus servicios.

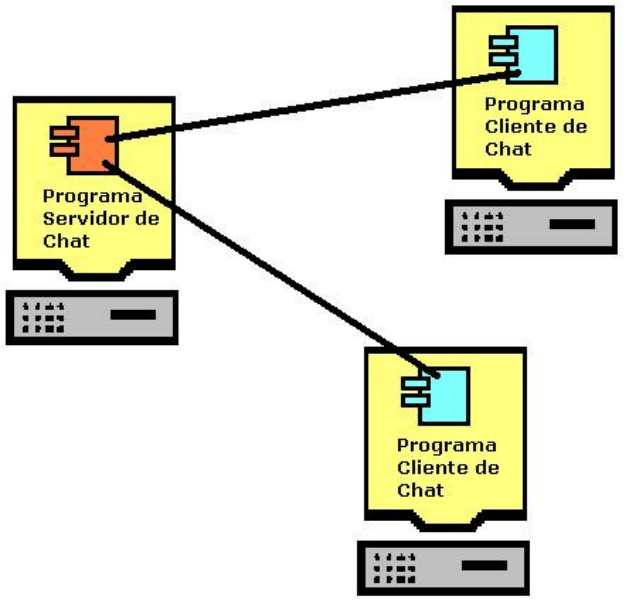
En esta arquitectura, la aplicación está compuesta por varios programas repartidos entre servidor y clientes. Cada programa se ejecuta en uno solo de los equipos y a su vez está compuesto por uno o varios componentes, cada uno de los cuales desempeña una función específica en el contexto de toda la aplicación.

Arquitectura Cliente-Servidor (continuación):

Ejemplo: Pensemos en una aplicación de ***chat*** sencilla. Dicha aplicación cuenta con:

- **Servidor**: En este equipo se ejecuta el programa servidor de chat, que centraliza el manejo de mensajes, así como la persistencia de los mismos.
- **Clientes**: En estos equipos se ejecuta el programa cliente, que se conecta con el servidor para enviar y recibir mensajes.

El programa servidor se ejecuta en un solo equipo, mientras que el programa cliente lo hace en cada uno de los equipos cliente.



Arquitectura Cliente-Servidor (continuación):

En el ejemplo del chat existen dos tipos de programas; el programa servidor y el programa cliente. El primero corre exclusivamente en el equipo servidor. Del segundo se ejecuta una copia en cada equipo cliente.

El programa servidor está formado por varios componentes, como ser el componente encargado de recibir y procesar los mensajes o el componente encargado de persistirlos. A su vez, el programa cliente está formado por el componente de interfaz gráfica para el usuario o el componente encargado de conectarse al servidor.

Cada programa está compuesto por uno o varios componentes, cada uno de los cuales está encargado de desempeñar una función específica dentro de toda la aplicación de chat. Obsérvese que además esta arquitectura **no** depende del tipo de red usada.

Arquitectura N-Tier:

Una arquitectura N-Tier consiste en:

- ***Uno o varios*** equipos denominados ***Servidores***. Brindan servicios a los Clientes residentes en otros equipos.
- ***Uno o varios*** equipos denominadas ***Clientes***, las cuales se comunican remotamente con distintos servidores.
- Algunos de los Servidores pueden a su vez ser Clientes de otros Servidores.

En esta arquitectura puede haber ***más de un*** Servidor, no todos brindan necesariamente los mismos servicios. Un mismo Cliente puede comunicarse con ***varios*** Servidores. No todos los Clientes acceden necesariamente a los mismos Servidores. Incluso puede suceder que algún equipo sea Servidor y Cliente a la vez.

UNIVERSIDAD DE LA EMPRESA

Arquitectura N-Tier (continuación):

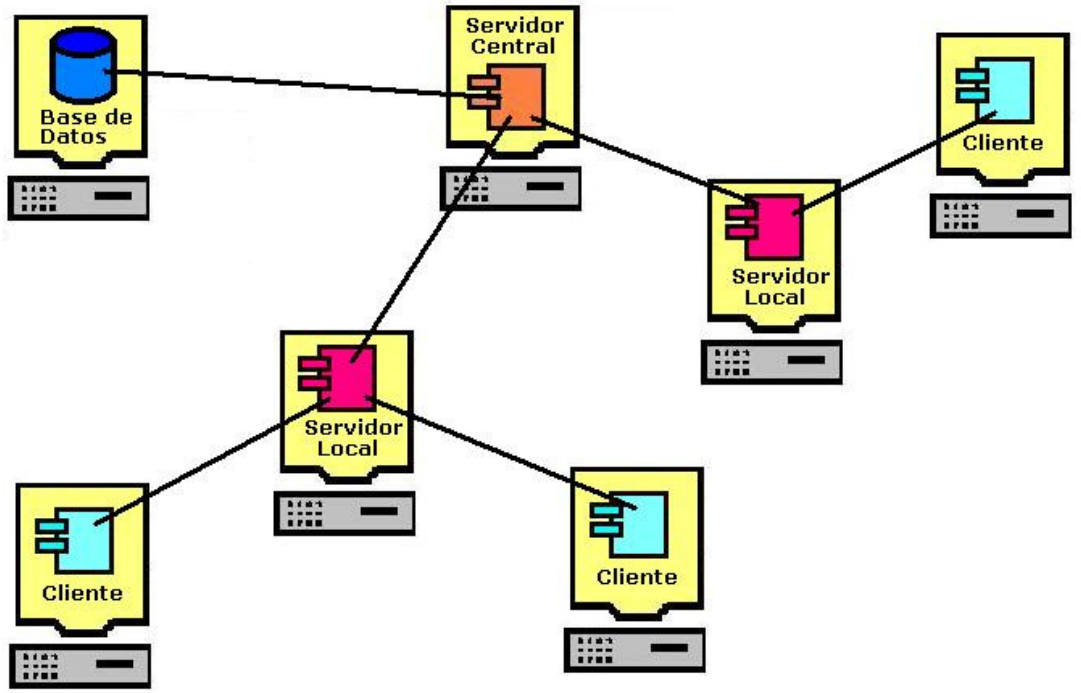
Ejemplo: Pensemos en una empresa que tiene su sede en Montevideo y distintas sucursales en el Interior. La empresa utiliza una aplicación con las siguientes características:

- En Montevideo residen la Base de Datos y el Servidor central.
- En cada sucursal del Interior hay un Servidor local que solicita servicios por Internet al Servidor central en Montevideo.
- Los empleados que trabajan en cada sucursal acceden al Servidor local mediante programas Cliente que corren en sus máquinas personales.

En esta aplicación existen dos tipos distintos de Servidores (central y local). Los servidores locales son a su vez clientes del servidor central. Además en este ejemplo cada cliente local se conecta a un único servidor (existen otros ejemplos donde puede no ser así).

Arquitectura N-Tier (continuación):

Este es un ejemplo de una arquitectura ***4-tier***. Esto es así porque existen ***4 niveles*** distintos de equipos.



Los componentes de cada tipo de equipo cumplen funciones que son propias y distintas de las que se ejecutan en los restantes tipos de equipos. Nótese que **no** es la cantidad total de equipos, sino la cantidad de **niveles** de equipos lo que define la arquitectura.

UDE
UNIVERSIDAD DE LA EMPRESA

Arquitectura N-Tier (continuación):

Observaciones:

- Tanto la cantidad como los tipos de Servidores y Clientes que pueden existir para una aplicación determinada, así como sus estrategias de interacción depende de cada problema concreto.
- En consecuencia, habrá aplicaciones que sigan una arquitectura **N-tier** más sencilla y otras que sigan una más compleja (notar la diferencia con **Cliente-Servidor**).
- La arquitectura Cliente-Servidor puede verse como un caso sencillo de una arquitectura **2-tier**.
- Pueden existir una o mas **Bases de Datos** repartidas entre toda la aplicación. Dependiendo del problema, las mismas podrán residir tanto en **clientes** como en **servidores**.

UNIVERSIDAD DE LA EMPRESA

Arquitectura lógica de una aplicación:

La arquitectura ***lógica*** de una aplicación se refiere a la manera en que los distintos componentes de la misma ***interactúan*** entre sí. Es decir, a la manera en que se comunican y se pasan información.

Esta arquitectura es **independiente** de la arquitectura física de la aplicación. Es decir, para su definición **no** interesa si la arquitectura física es Standalone, Cliente-Servidor o N-Tier.

La arquitectura lógica de una aplicación surge del principio de ***Separación en Capas*** estudiado en cursos anteriores. La cantidad de capas que se use al definir los distintos componentes de la aplicación junto con la naturaleza de tales componentes es lo que define finalmente su arquitectura lógica.

UNIVERSIDAD DE LA EMPRESA

Arquitectura lógica de una aplicación (continuación):

El diseño de aplicaciones persigue cualidades como **eficiencia** y **robustez**, pero también otras como **reusabilidad**, **extensibilidad** o **anticipación al cambio**.

Para lograr tales cualidades, el principio de ***Separación en Capas*** establece que los componentes de toda aplicación se diseñen de acuerdo a las siguientes tres capas:

- Capa ***Gráfica*** (también llamada de Presentación)
- Capa ***Lógica*** (también llamada de Negocio)
- Capa de ***Persistencia*** (también llamada de Acceso a Datos)

Si bien el **ideal** es que la arquitectura lógica se defina siempre en función de estas tres capas, existen aplicaciones en las que se trabaja con menos capas. Dependiendo de la aplicación, los costos en términos de las cualidades pueden ser mayores o menores.

UNIVERSIDAD DE LA EMPRESA

Arquitectura lógica de tres capas:

El principio de ***Separación en Capas*** establece las siguientes responsabilidades para cada capa:

- Capa ***Gráfica***: Se encarga exclusivamente de la interacción con los usuarios de la aplicación. Brinda la interfaz de usuario (IU).
- Capa ***Lógica***: Se encarga exclusivamente de la resolución de la lógica de los requerimientos funcionales (casos de uso) de la aplicación (o del subsistema en cuestión).
- Capa ***Persistencia***: Se encarga exclusivamente del respaldo y mantenimiento de la información manejada por el sistema (ya sea en **Bases de Datos** o en otro mecanismo de persistencia).

Este principio también establece que las capas ***gráfica*** y de ***persistencia*** **nunca** interactúen directamente, sino que lo hagan **siempre** a través de la capa ***lógica***.

UNIVERSIDAD DE LA EMPRESA

Arquitectura lógica de una y dos capas:

Como se dijo antes, el **ideal** es que la arquitectura lógica se defina en **tres** capas. No obstante, existen aplicaciones cuya arquitectura lógica consta de **una** o de **dos** capas.

- ***Arquitecturas de una capa***: Las tres capas se fusionan en una sola. Estas aplicaciones acceden desde la interfaz de usuario directamente al mecanismo de persistencia. La lógica de los requerimientos se resuelve en forma conjunta con el respaldo y la presentación de los datos al usuario.

- ***Arquitecturas de dos capas***: La capa gráfica se separa y se fusionan las capas lógica y de persistencia. Estas aplicaciones separan la presentación al usuario del procesamiento interno de los datos, que resuelve la lógica de los requerimientos en conjunto con el respaldo de los datos.

Ejemplos de Arquitecturas lógicas y físicas:

Ejemplo: Pensemos en una aplicación de escritorio que gestiona la contabilidad de una pequeña empresa. La misma está desarrollada en ***C++*** y corre sobre ***Linux***. Los datos son persistidos en una estructura de archivos de texto en el File System. Existe un módulo encargado de manejar la interacción con el usuario a través de consola. Los datos ingresados son pasados a otro módulo que se encarga de resolver los requerimientos de la aplicación. Existe un tercer módulo que se encarga de respaldar en disco toda la información una vez ejecutado el requerimiento.

- **Describir cuántos *componentes* posee esta aplicación.**
- **Describir la *arquitectura física* de esta aplicación.**
- **Describir la *arquitectura lógica* de esta aplicación.**

UNIVERSIDAD DE LA EMPRESA

Ejemplos de Arquitecturas lógicas y físicas (continuación):

Ejemplo: Pensemos en una aplicación Web para pedidos on-line de un delivery de empanadas. La empresa tiene un equipo con una instalación del Servidor Web ***Tomcat*** y otro PC con una instalación del DBMS ***PostgreSQL***. Ambos están conectadas en una red LAN, pero sólo el equipo del Servidor Web tiene salida a Internet. La aplicación consiste en una página JSP en la que el usuario elige los sabores de las empanadas e ingresa los datos del envío. La página JSP establece una conexión con la base de datos, registra en ella el pedido y forwardea a otra página JSP que confirma el pedido.

- **Describir cuántos *componentes* posee esta aplicación.**
- **Describir la *arquitectura física* de esta aplicación.**
- **Describir la *arquitectura lógica* de esta aplicación.**

UNIVERSIDAD DE LA EMPRESA

Arquitectura lógica versus Arquitectura física:

Como se dijo antes, no existe relación entre la ***arquitectura lógica*** y la ***arquitectura física*** de una aplicación. Esto permite que existan sistemas con múltiples combinaciones entre ambas arquitecturas.

Por ejemplo:

- Aplicaciones que poseen una arquitectura ***física*** **standalone** y una arquitectura ***lógica*** de **tres** capas.
- Aplicaciones que poseen una arquitectura ***física*** **distribuida** (tanto Cliente-Servidor como N-Tier) que sin embargo poseen una arquitectura ***lógica*** de **una** sola capa.
- Etcétera (existen múltiples combinaciones).

Si bien el **ideal** es que la arquitectura ***lógica*** sea de **tres** capas, cada problema concreto determinará qué combinación de ambas arquitecturas aplicar (lo que tendrá sus pros y contras).

UNIVERSIDAD DE LA EMPRESA

Arquitectura lógica versus Arquitectura física (continuación):

La ***arquitectura física*** de una aplicación suele estar determinada de antemano por la realidad del problema (**requerimientos no funcionales**) y suele definirse durante la concepción del proyecto.

Sin embargo, la ***arquitectura lógica*** de la aplicación suele definirse durante la fase de ***diseño*** y se define en base a las funcionalidades (**requerimientos funcionales**) que el sistema debe brindar.

Las **Bases de Datos** constituyen un mecanismo de ***persistencia***, y la capa de persistencia forma parte de la ***arquitectura lógica*** de la aplicación. Por esta razón, los temas de este curso se centrarán en el estudio de esta arquitectura más que en la ***arquitectura física***.

En capítulos siguientes se estudiará en detalle el acceso a **Bases de Datos** (y a la persistencia en general) en aplicaciones con una ***arquitectura lógica*** de **una**, **dos** y **tres** capas.