Bases de Datos

Tema 05. Diseño y Desarrollo de Aplicaciones de Bases de Datos

Marta Elena Zorrilla Pantaleón

Rafael Duque Medina

DPTO. DE MATEMÁTICAS, ESTADÍSTICA Y COMPUTACIÓN

Tabla de contenido

Referencias

- Cosentino, Christopher. Guía esencial de PHP. Prentice Hall, 2001.
- Fisher Maydene; Ellis Jon; Bruce, Jonathan. JDBC(TM) API Tutorial and Reference (3° Ed.). Addison-Wesley, 2003.
- Melton, Jim, Eisenberg, Andrew. SQL y Java : guía para SQLJ, JDBC y tecnologías relacionadas. Ra-Ma, 2001.
- Schafer, Steven M. HTML, XHTML y CSS. Anaya Multimedia, 2010.
- Villapecellín Cid, M. Arquitecturas de red multicapa: conexión de bases de datos. Ra-Ma, cop. 2005.

Concepto de aplicación

- Una **aplicación** es un programa informático que realiza una tarea.

- Una **aplicación de base de datos** es un programa que utiliza los datos de un sistema de gestión de bases de datos (SGBD).

- La mayoría de la aplicaciones de base de datos presentan, introducen y actualizan datos en la base de datos.

Lenguajes

- Desde los años 80 hasta nuestros días se han utilizado diversos lenguajes para construir aplicaciones de acceso a datos:
  - En los años 80 y principios de los 90 se utilizaron lenguajes de 3ª generación, en particular, el Cobol y el C con instrucciones SQL embebidas (estándar SQL/CLI). Estos lenguajes ofrecían gran flexibilidad y rendimiento.
  - A continuación las casas del productos de BDs como Oracle, Informix, etc. desarrollaron herramientas vinculadas a sus gestores que facilitaban la construcción de aplicaciones de gestión de datos, haciendo uso de los catálogos de la BD. Estos se denominaron lenguajes de 4ª generación (4GL) y se caracterizan por ser más declarativos y orientados a datos. Ej. Informix 4GL, Ingres 4GL, Oracle Reports, etc.

Lenguajes: ej. SQL embebido (y 2)

```
#include <stdio.h>
#include <stdlib.h>
main(){
EXEC SQL BEGIN DECLARE SECTION;
  /*Sección para declarar variables */
EXEC SQL END DECLARE SECTION;
EXEC SQL connect to 'NombreBD' as 'conbd' WITH CONCURRENT TRANSACTION;
printf("Conectando a BD: SQLCODE = %d\n",SQLCODE);
if (SQLCODE != 0)   {
   printf("Error conexion: SQLCODE = %d\n",SQLCODE);
    exit();   }
EXEC SQL set connection 'conbd';
if (SQLCODE != 0)   {
   printf("Error en SET conexión a BD");
   EXEC SQL disconnect ALL;
   exit();   }
EXEC SQL update TABLA set CAMPO_FECHAANUL=null  where CAMPO_FECHAANUL is not null;
if (strncmp(SQLSTATE,"00",2) != 0)
  {
   printf(" Error Update: %d - %s\0",SQLCODE,sqlca.sqlerrm);
  EXEC SQL disconnect ALL;
   exit();
  }
EXEC SQL disconnect ALL; }
```

Lenguajes (y 3)

- Los lenguajes 4GL de aquella época se caracterizaban por:
  - Estar bastante ligados a los gestores
  - Ofrecer bastante funcionalidad por defecto lo que reducía tiempos de desarrollo (generadores de informes, pantallas y menús) y el nº de errores
  - Ofrecer entornos amigables, muy orientados a los datos y al proceso de aplicaciones transaccionales
  - Aunque presentaban también ciertas limitaciones ya que estaban ligados al catálogo del gestor con el que trabajan, que en aquella época era propietario (no se habían definido las vistas Information Schema) lo que hacía difícil la integración con otras aplicaciones, que generalmente se construían con 3GL. Además el proceso de depuración era más costoso que en 3GL.

Lenguajes: ejemplo 4GL

```
database DBDB
screen
{



                              TABLA 1



            Codigo:[a    ]

            Nombre:[b                    ]
|




}
end
tables
tabla1
attributes
a    = tabla1.codigo  ,REVERSE,AUTONEXT,UPSHIFT,
       COMMENTS="Codigo...           ";
b    = tabla1.nombre  ,REVERSE,AUTONEXT,UPSHIFT,
       COMMENTS="Descripcion...           ";
end
instructions
delimiters"  "
end
```

Formato de pantalla en I-4GL

```
FUNCTION dar_alta_tabla1()
DEFINE t1 SMALLINT
DISPLAY "Pulsar <ESC> para aceptar los Datos     " AT 1,1
DISPLAY "Pulsar <F1> para volver al menu " AT 2,1
# leemos de pantalla
INPUT BY NAME p_tabl.* WITHOUT DEFAULTS
      ON KEY (F1)
        LET t1 = 1
        EXIT INPUT
      AFTER FIELD codigo
        IF p_tabl.codigo IS NULL THEN
            ERROR "ERROR: Dato Obligatorio." ATTRIBUTE(YELLOW)
            NEXT FIELD codigo
        END IF
END INPUT
#si no se pulsa F1 ni ctrl-supr , se inserta
   IF INT_FLAG = FALSE AND t1 = 0 THEN
      BEGIN WORK
      INSERT INTO tabla1  VALUES (p_tabl.*)
      IF estado != 0 THEN
               ROLLBACK WORK
                 MESSAGE "Alta NO efectuada"  ATTRIBUTE(YELLOW)
                  RETURN
      END IF
      COMMIT WORK
      MESSAGE "Alta efectuada" ATTRIBUTE(YELLOW)
   END IF
END FUNCTION
```

Función alta en I-4GL

Lenguajes (y 5)

- En la actualidad se realizan los desarrollos de aplicaciones utilizando **herramientas RAD** (desarrollo rápido de aplicaciones) o IDEs (Integrated Development Environment), los cuales automatizan algunas tareas haciéndolas transparentes para el programador como generar formularios o informes (.Net, WebSphere, PowerBuilder, NetBeans, etc)
- Sin embargo, es necesario esforzarse en emplear métodos de diseño, pues en caso contrario nos arriesgamos a crear aplicaciones monolíticas de escaso valor en cuanto a reutilización. → generar clases básicas (framework)
- Su facilidad de uso en la parte de desarrollo de la interfaz de usuario no favorece la programación cuidadosa de la aplicación, en términos de diseño.
- No es necesario para el programador comprender cómo funcionan realmente estas herramientas, o sus jerarquías de clases.
- El desarrollo de una interfaz se traduce en "montarla“ empleando diferentes componentes.
  - Se usan distintos objetos y se pone código en los eventos.

Aplicaciones interactivas: Definiciones

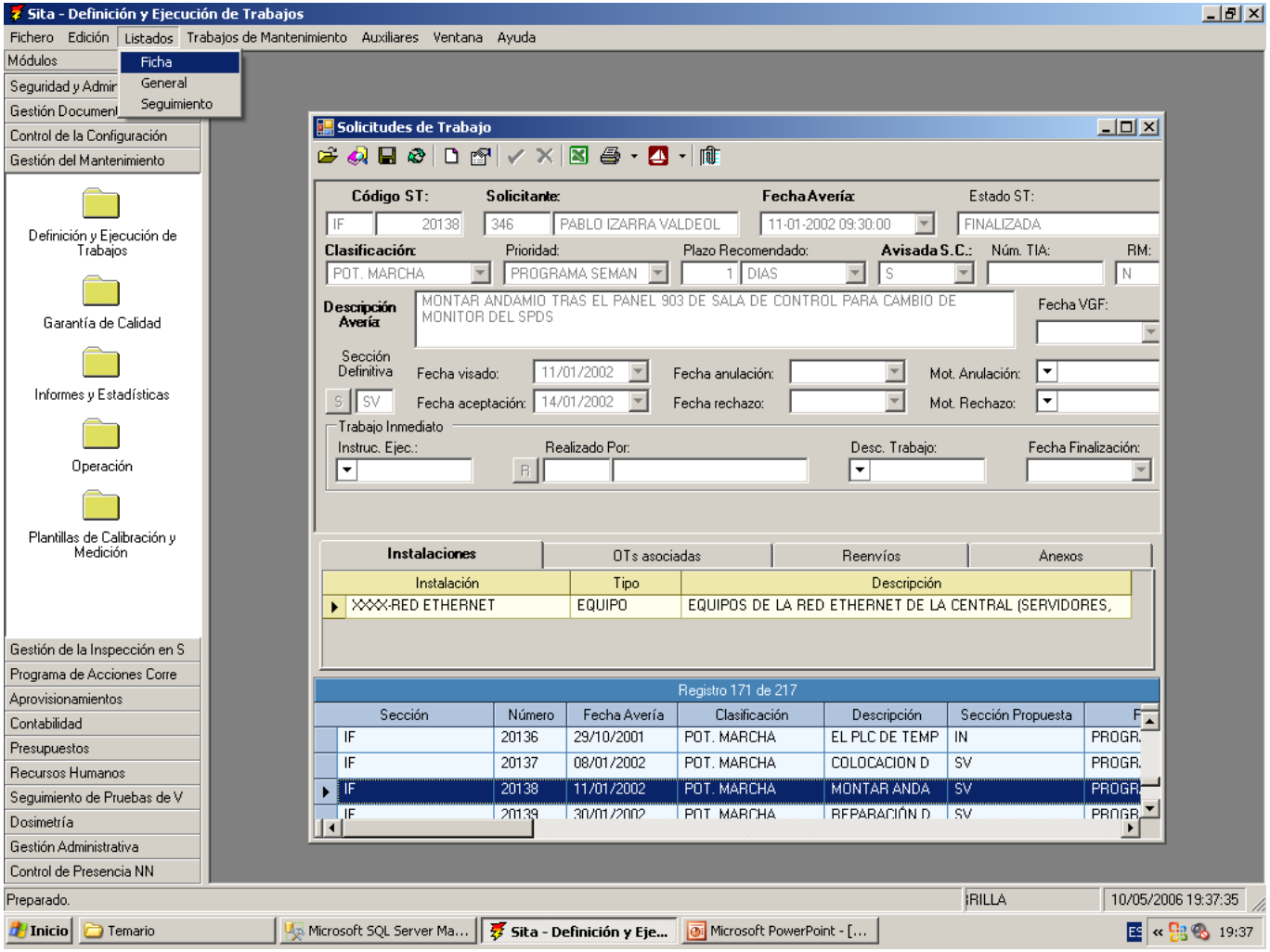
- Existen muchos tipos de aplicaciones pero en general el acceso y manipulación de datos en el que intervienen humanos se realiza a través de aplicaciones interactivas.
  - Interacción: Todos los intercambios que suceden entre la persona y el ordenador
  - Interfaz de usuario: Son las partes del sistema con las que el usuario entra en contacto física y cognitivamente
    - Interacción física (teclado, ratón, pantalla...)
    - Interacción cognitiva (lo que se presenta al usuario debe ser comprensible para él)
  - Usabilidad: Es el grado en el que un producto puede ser utilizado por los usuarios para conseguir sus objetivos con efectividad, eficiencia y satisfacción

Elementos de las aplicaciones

- Las aplicaciones modernas consisten generalmente en una interfaz gráfica de usuario con menús, barras de herramientas, cuadros de diálogo y ventanas que muestran los datos de la aplicación.

- Hay tres tipos de elementos:
  - los formularios,
  - los informes,
  - los gráficos.

Ejemplo


Sita - Definición y Ejecución de Trabajos
Fichero Edición Listados Trabajos de Mantenimiento Auxiliares Ventana Ayuda
Ficha
General
Seguimiento
Módulos
Seguridad y Admin
Gestión Document
Control de la Configuración
Gestión del Mantenimiento
Definición y Ejecución de Trabajos
Garantía de Calidad
Informes y Estadísticas
Operación
Plantillas de Calibración y Medición
Gestión de la Inspección en S
Programa de Acciones Corre
Aprovisionamientos
Contabilidad
Presupuestos
Recursos Humanos
Seguimiento de Pruebas de V
Dosimetría
Gestión Administrativa
Control de Presencia NN
Solicitudes de Trabajo
Código ST:
Solicitante:
Fecha Avería:
Estado ST:
IF 20138 346 PABLO IZARRA VALDEOL 11-01-2002 09:30:00 FINALIZADA
Clasificación:
Prioridad:
Plazo Recomendado:
Avisada S.C.:
Núm. TIA:
RM:
POT. MARCHA PROGRAMA SEMAN 1 DIAS S N
Descripción Avería:
MONTAR ANDAMIO TRAS EL PANEL 903 DE SALA DE CONTROL PARA CAMBIO DE MONITOR DEL SPDS
Fecha VGF:
Sección Definitiva
Fecha visado: 11/01/2002
Fecha anulación:
Mot. Anulación:
S SV
Fecha aceptación: 14/01/2002
Fecha rechazo:
Mot. Rechazo:
Trabajo Inmediato
Instruc. Ejec.:
Realizado Por:
Desc. Trabajo:
Fecha Finalización:
Instalaciones
OTs asociadas
Reenvíos
Anexos
Instalación Tipo Descripción
XXXX-RED ETHERNET EQUIPO EQUIPOS DE LA RED ETHERNET DE LA CENTRAL (SERVIDORES,
Registro 171 de 217
Sección Número Fecha Avería Clasificación Descripción Sección Propuesta
IF 20136 29/10/2001 POT. MARCHA EL PLC DE TEMP IN PROGR
IF 20137 08/01/2002 POT. MARCHA COLOCACION D SV PROGR
IF 20138 11/01/2002 POT. MARCHA MONTAR ANDA SV PROGR
IF 20139 30/01/2002 POT. MARCHA REPARACIÓN D SV PROGR
Preparado.
10/05/2006 19:37:35
Inicio
Temario
Microsoft SQL Server Ma...
Sita - Definición y Eje...
Microsoft PowerPoint - [...
19:37

Formularios

- Un formulario es una ventana (o página Web) que consta de campos /o rejillas donde presenta los datos y permite al usuario interactuar con ellos.

- El formulario proporciona una buena forma de ver y manipular la información que contiene la aplicación.

- Su usabilidad depende del diseño.

Diseño de formularios

- Los formularios **sencillos** se corresponden con un tipo de registro, que es una serie de campos de datos (una única fila de datos). Un formulario puede mostrar uno o varios registros a la vez.
- Formularios **maestro-detalle**, que dividen el formulario en un registro maestro y varios registros que dependen de él (p.ej. el pedido con sus líneas).
- Los formularios generalmente incluyen botones de acción (consulta, inserción, actualización, borrado, desplazamiento, etc.)
- Los generadores de aplicaciones (Wizard) suministran estas capacidades (caso Access).
- Las aplicaciones de las grandes organizaciones generalmente construyen su ***framework*** (clases en java o C#) particular, utilizando los generadores para construir prototipos.

Ejemplo


EncabezadoPedido
GESTION DE PEDIDOS
Situación: Recepcionado
Nº Pedido 2
Fecha pedido 10/06/2002
Fecha entrega 15/08/2002
Proveedor P002
Nombre Zar Luna, Ana
CIF A48162311
Dirección Ercilla 22, 1º
Tfno. (34) 947 865 456
Localidad Bilbao
C. postal 48002
Fax (34) 947 865 456
Nº Línea Artículo Unidades Precio unitario Descuento Total línea
1 SILLA ERGONOMICA MOI 3 120,00 € 2 352,80 € Entradas
2 ARMARIO DIPLOMATIC 3 300,00 € 3 873,00 € Entradas
3 SILLA ERGONOMICA MOI 5 120,00 € 0 600,00 € Entradas
Importe neto: 1.825,8 €
Registro: 2 de 26

Directrices de usabilidad para el diseño de formularios

- Dar un título al formulario que exprese claramente su función.
- Las instrucciones han de ser breves y comprensibles.
- Hacer grupos lógicos de campos y separarlos.
  - P.ej: nombre, primer y segundo apellido es un grupo lógico.
- Aspecto ordenado alineando los campos y las etiquetas.
- Las etiquetas de los campos deben usar terminología familiar.
- Ser consistente en el uso de los términos, es decir, usar siempre las mismas palabras para los mismos conceptos.
- El tamaño visible del campo debe corresponderse con la longitud del contenido que ha de introducir el usuario.
- Permitir el movimiento del cursor por medio del teclado y no solo con el ratón.

Directrices de usabilidad para el diseño de formularios (cont.)

- Permitir que el usuario pueda corregir con libertad los caracteres que ha introducido en los campos.
- En donde sea posible, impedir que el usuario introduzca valores incorrectos.
  - Por ejemplo, impedir que introduzca caracteres alfabéticos en campos que solo admiten valores numéricos.
- Si introduce valores incorrectos, indicar en un mensaje cuales son los correctos.
- Avisar cuanto antes al usuario si ha introducido valores incorrectos. Si es posible, no esperar a que haya rellenado el formulario totalmente.
- Marcar claramente los campos opcionales.
- Si es posible, colocar explicaciones o la lista de los valores válidos al lado de los campos.
- Dejar clara la acción que debe hacer el usuario al terminar de rellenar el formulario

Directrices de usabilidad para el diseño de formularios (cont.)

- Introducción de datos
  - Reducir al máximo el número de datos a introducir por el usuario. Pedir solo la información necesaria y autocompletar la que se pueda inferir
  - Los combos desplegables frecuentemente dificultan más que ayudan a completar los formularios.
- Formularios en varios pasos (generalmente Web)
  - Si el formulario es muy largo, dividir en pasos e indicar el número de pasos y la situación en el proceso, por ejemplo, "paso 2 de 4".
  - Posibilitar los movimientos, "Anterior" y "Siguiente", dentro del formulario y la acción de "Cancelar" el proceso.
  - Evitar abrir páginas en otras ventanas de navegador (Web)

Prototipado

- El prototipado hace referencia al proceso de construcción rápido de un "sistema preliminar", al que quizá le faltan algunas características, pero con un funcionamiento sustancialmente completo, mediante el cual puede demostrarse su funcionalidad básica.

- Ventajas:
  - El usuario entiende qué se va a hacer y tiene más facilidad para transmitir las ideas
- Inconvenientes:
  - El peligro es que los usuarios piensan que el prototipo es suficientemente bueno.
  - A menudo estos prototipos no funcionan bien, pueden ser poco seguros y carecer de las características de facilidad de uso que los usuarios pueden no saber qué necesitan.
  - El programador se puede encontrar con dificultades para incorporar nuevas características por su poca flexibilidad.
  - También pueden acarrear problemas para justificar el tiempo de desarrollo

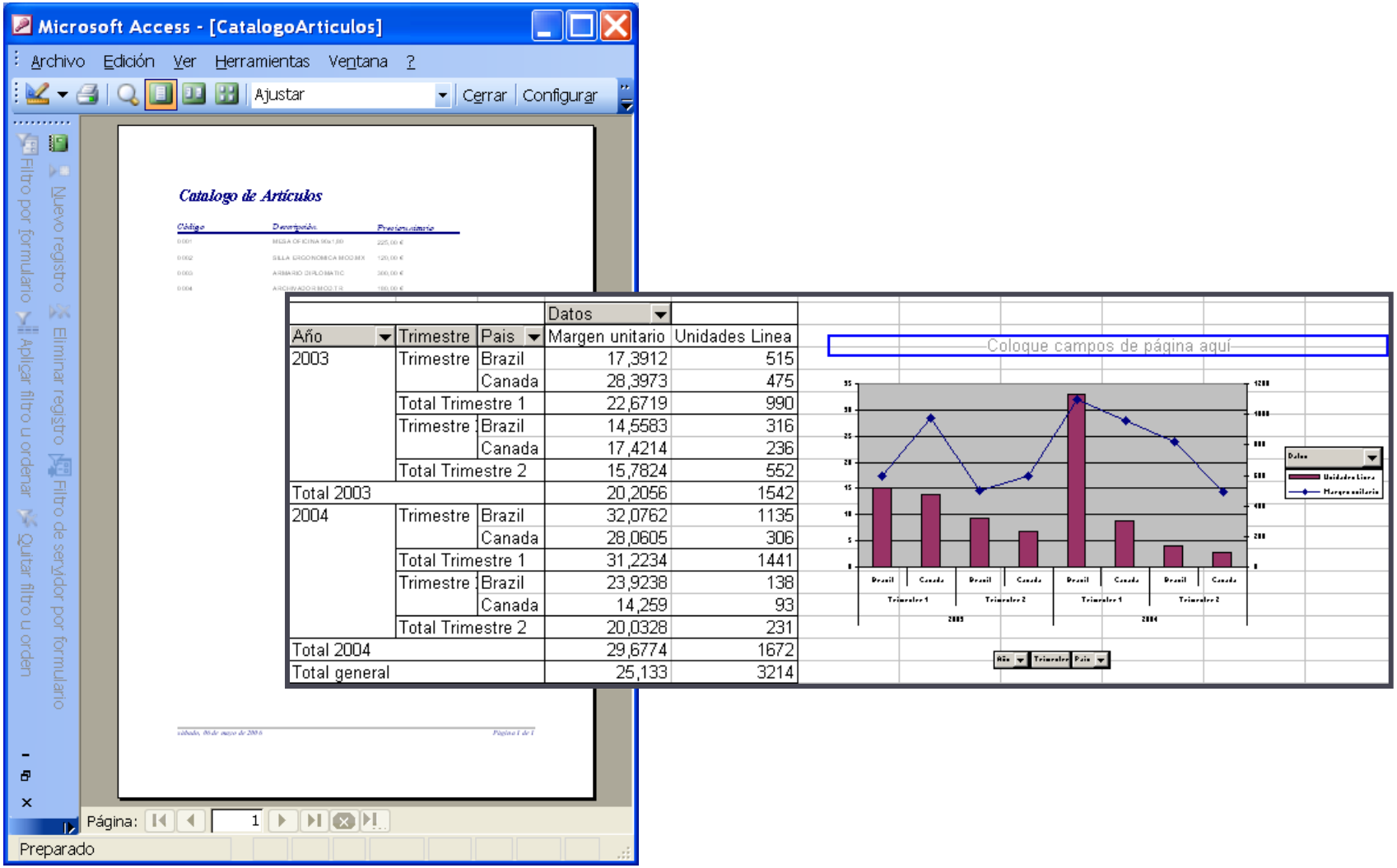
Informes

- Un informe es una presentación de datos orientada a su visualización en pantalla o impresión en papel. (facturas, recibos, etc.)

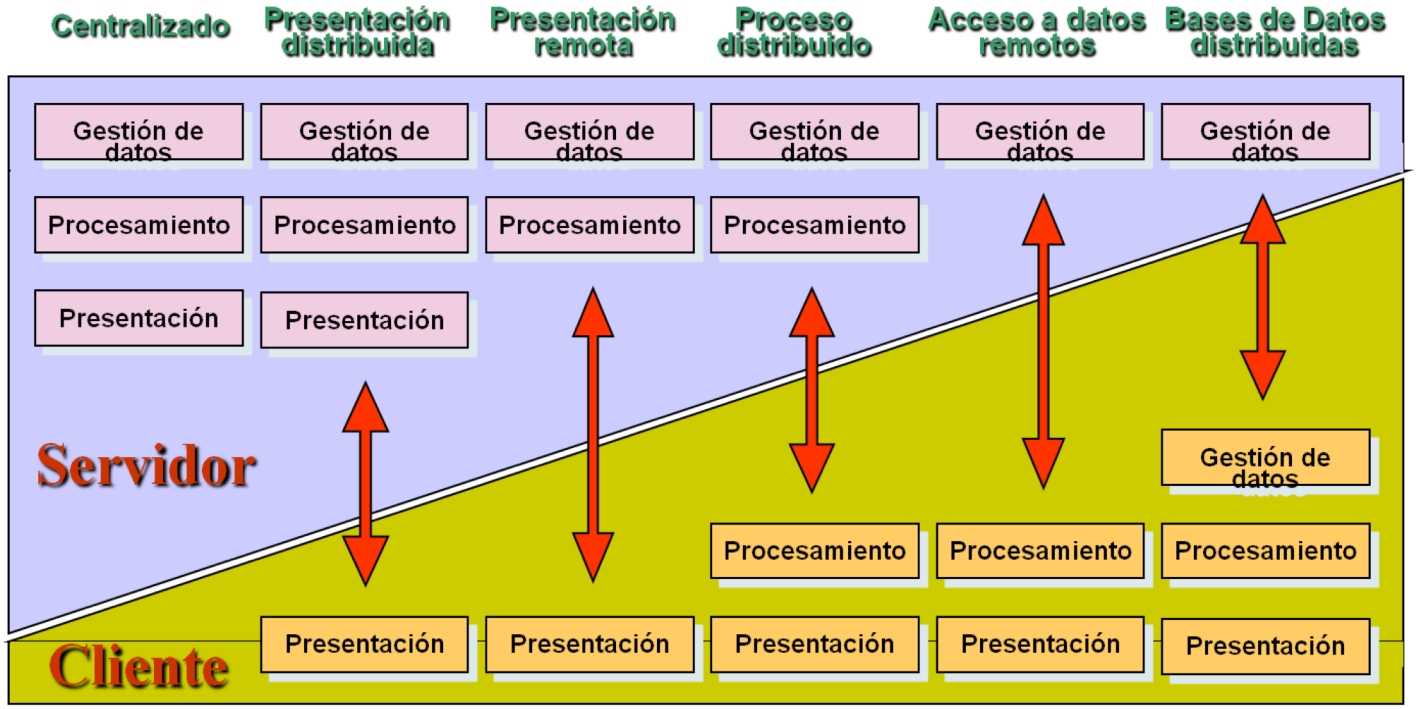
- No permite la manipulación de datos aunque tiene herramientas que permiten mover, rotar, imprimir, etc. las páginas de las que consta el informe.

- Los informes pueden incluir gráficos que ayuden a interpretar su contenido.

Ejemplo


Microsoft Access - [CatalogoArticulos]
Archivo Edición Ver Herramientas Ventana ?
Ajustar
Cerrar
Configurar
Catalogo de Artículos
Página:
Preparado
Datos
Año
Trimestre
Pais
Margen unitario
Unidades Linea
2003 Trimestre Brazil 17,3912 515
Canada 28,3973 475
Total Trimestre 1 22,6719 990
Trimestre Brazil 14,5583 316
Canada 17,4214 236
Total Trimestre 2 15,7824 552
Total 2003 20,2056 1542
2004 Trimestre Brazil 32,0762 1135
Canada 28,0605 306
Total Trimestre 1 31,2234 1441
Trimestre Brazil 23,9238 138
Canada 14,259 93
Total Trimestre 2 20,0328 231
Total 2004 29,6774 1672
Total general 25,133 3214
Coloque campos de página aquí

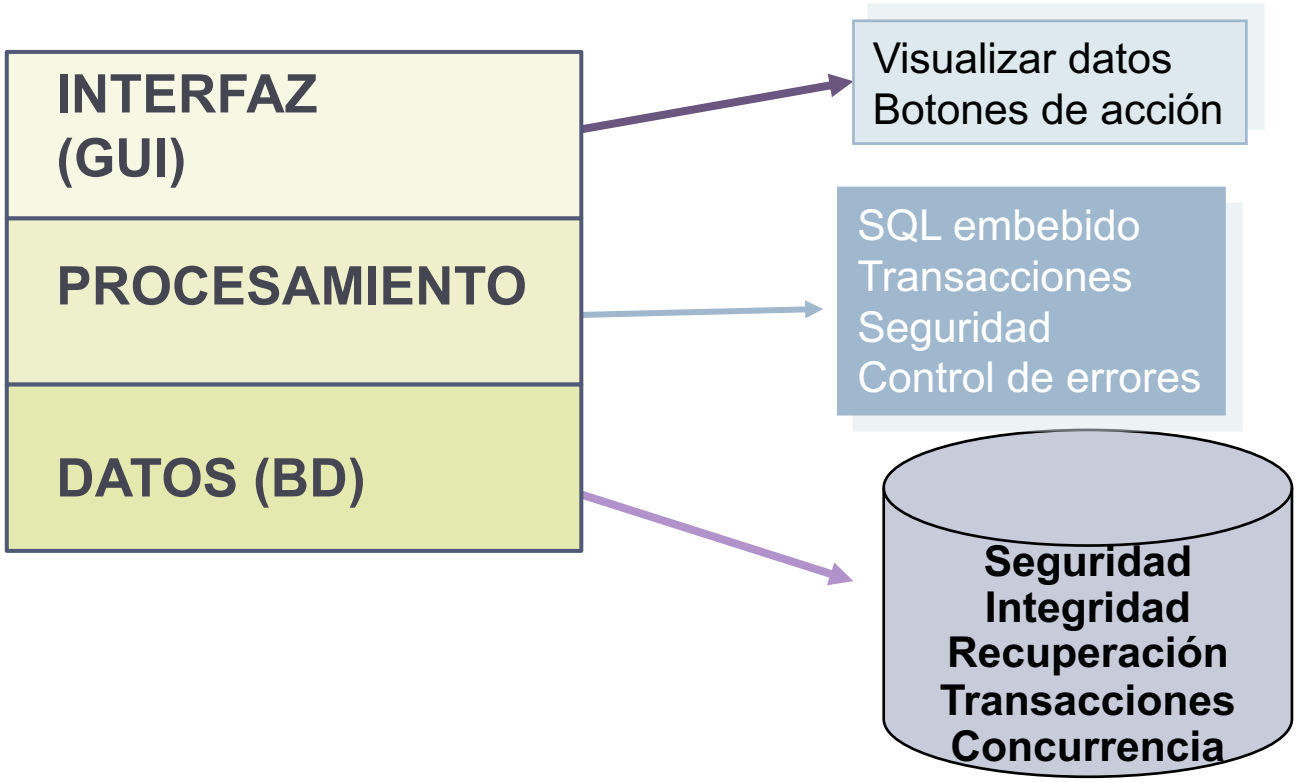
Arquitecturas de la aplicaciones de BD



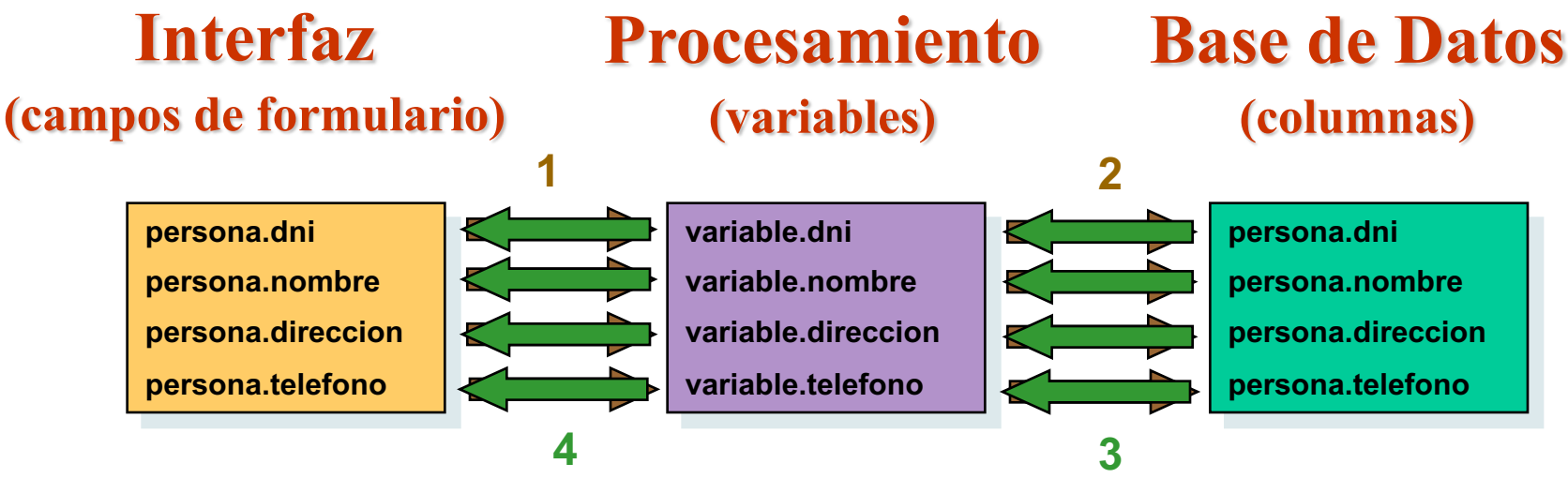
Desde los años 80 hasta la actualidad la arquitectura de las aplicaciones se ha ido modificando pasando desde una arquitectura centralizada (todo en el host) a una distribuida, en el que se reparte el aplicativo en varias máquinas (cliente y servidores) aprovechando así las capacidades gráficas y de cómputo de los equipos cliente

Arquitectura de la aplicaciones de BD

- En la actualidad se desarrollan las aplicaciones en varias capas, siendo 3 el nº mínimo que responden a la capa de interfaz, la de lógica de negocio y la capa de datos



Flujo de datos en aplicaciones de BD



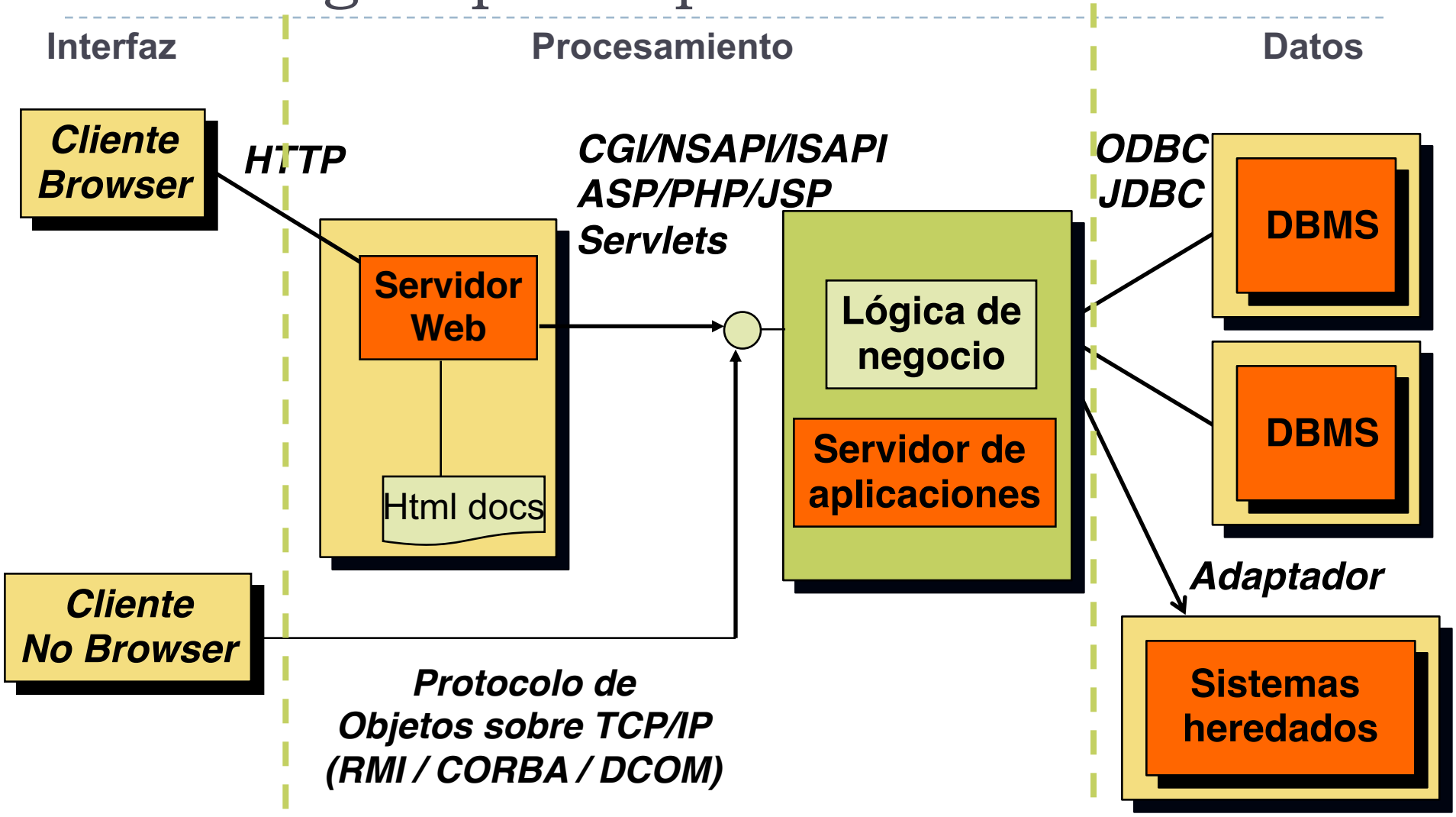
1.- Lectura de campos de la pantalla

2.- Instrucción SQL (INSERT, UPDATE, DELETE, SELECT sobre tablas o vistas, PROCEDIMIENTOS ALMACENADOS...)

3.- Resultado de la instrucción SQL

4.- Volcado a campos de la pantalla

Tecnologías para aplicaciones de BD



Clientes

- **Cliente pesado** (thick):
  - Menos requerimientos del servidor
  - Mejor desempeño multimedia
  - Más flexibilidad (interfaz y también procesamiento)
  - Mejor soporte de periféricos
  - Complejos de instalar y mantener

- **Cliente ligero** (thin):
  - Menores requisitos hardware
  - Lógica centralizada (sólo capa de interfaz)
  - Mayor seguridad en los datos
  - Fáciles de desplegar y mantener

Clientes ligeros (browser)

- **Ventajas**
  - Interface de usuario universal
  - Coste reducido
  - Acceso desde equipos heterogéneos
  - Acceso muy difundido
  - Requisitos de hardware mínimos
  - Alta disponibilidad
- **Inconvenientes**
  - Estándares de Browser y lenguaje HTML
  - Menos funcionalidades que las aplicaciones de escritorio → RIA (Rich Internet Applications)

Otros clientes ligeros

- Network computers
- Netbooks
- Teléfonos inteligentes (PDA)
- Dispositivo de Internet móvil (MID)

Cliente ligero - Tecnologías

- HTML / DHTML / CSS
- Lenguajes de script: JavaScript / VBScript
- Applet Java / Active X
- AJAX (Asynchronous JavaScript And XML):
  - Nuevo enfoque para aplicaciones web interactivas
  - Ejecución en el navegador del usuario
  - Comunicación asíncrona con el servidor cuando sea necesario

Clientes pesados -Protocolo de objetos

- **RMI.-** Remote Invocation Method.- Fue el primer framework para crear sistemas distribuidos de Java. El sistema de Invocación Remota de Métodos (RMI) de Java permite, a un objeto que se está ejecutando en una Máquina Virtual Java (VM), llamar a métodos de otro objeto que está en otra VM diferente. Esta tecnología está asociada al lenguaje de programación Java, es decir, que permite la comunicación entre objetos creados en este lenguaje.
- **DCOM.-** Distributed Component Object Model.- El Modelo de Objeto Componente Distribuido, esta incluido en los sistemas operativos de Microsoft. Es un juego de conceptos e interfaces de programa, en el cual los objetos de programa del cliente, pueden solicitar servicios de objetos de programa servidores en otros ordenadores dentro de una red. Esta tecnología esta asociada a la plataforma de productos Microsoft.
- **CORBA.-** Common Object Request Broker Architecture.- Tecnología introducida por el Grupo de Administración de Objetos OMG, creada para establecer una plataforma para la gestión de objetos remotos independiente del lenguaje de programación.

HTTP

- Hypertext Transfer Protocol (HTTP)
  - Protocolo Petición/Respuesta (El navegador realiza una solicitud HTTP y el servidor procesa la solicitud y después envía una respuesta HTTP)
  - Usa puerto 80 por defecto
  - Protocolo sin estado, cierra la sesión con la respuesta
  - HTTP 1.1 fue creado en Junio de 1999 por W3C (world wide web consortium) y el IETF (Internet Engineering Task Force)
  - 8 métodos o verbos para hacer peticiones:

Comando	Descripción
GET	Solicita el recurso ubicado en la URL especificada
HEAD	Solicita el encabezado del recurso ubicado en la URL especificada
POST	Envía datos al programa ubicado en la URL especificada
PUT	Envía datos a la URL especificada
DELETE	Borra el recurso ubicado en la URL especificada
TRACE	Permite al cliente conocer la situación de su petición
OPTIONS	Información sobre opciones de comunicación disponibles
CONNECT	Se utiliza para trabajar con proxies

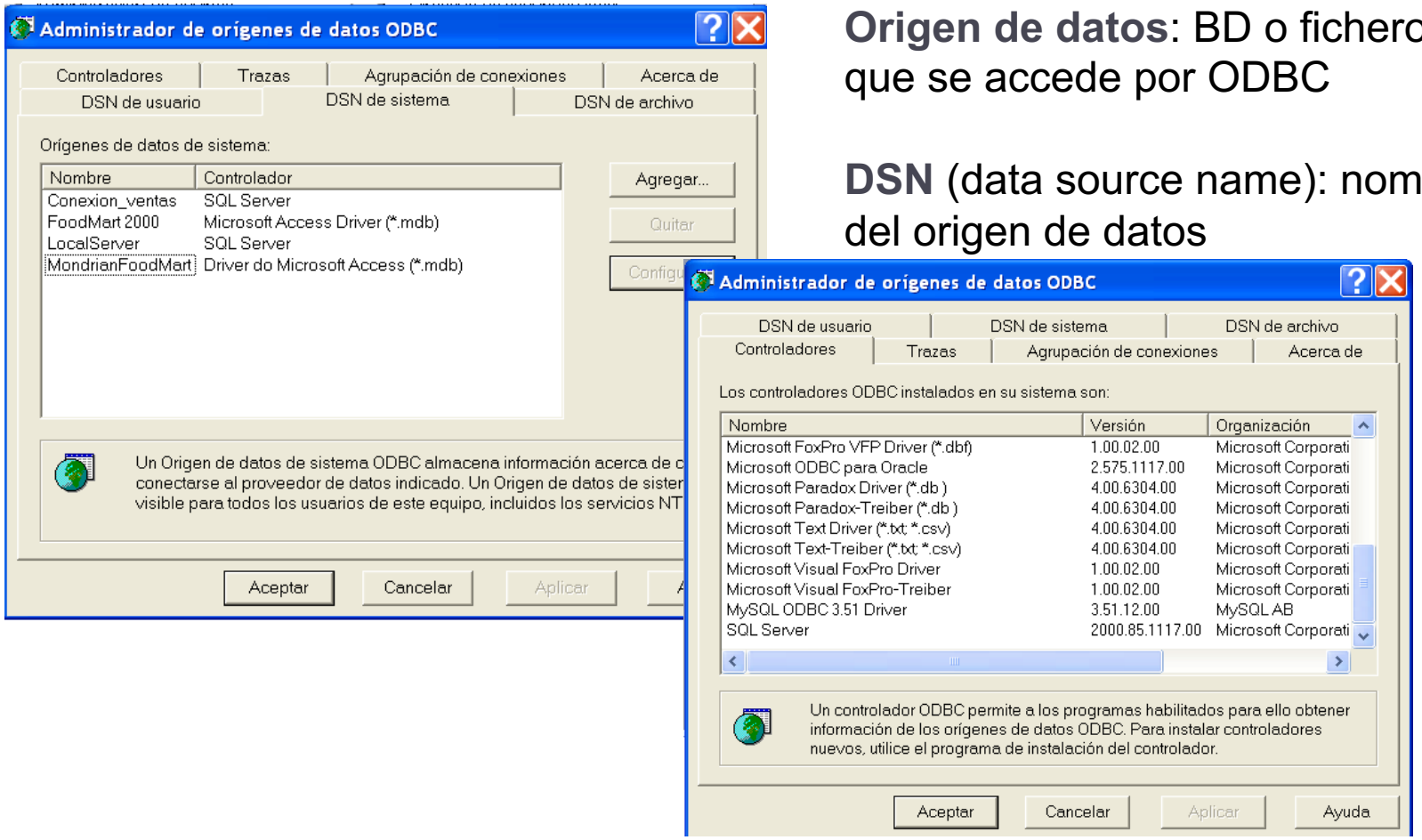
CGI

- Common Gateway Interface
  - Nace en 1993 (Rob McCool, John Franks, Ari Luotonen, George Phillips and Tony Sanders)
  - Estándar de facto para ejecutar programas de forma remota
    - http://www.dominio.com/ejemplo.cgi
  - El CGI es un programa compilado o interpretado que se ejecuta en el servidor bajo demanda del cliente
  - Problema de sobrecarga en el servidor
  - A partir del CGI nacen otras extensiones de servidor que ofrezcan mejor rendimiento y compartir recursos
- NSAPI: CGI Netscape
- ISAPI: CGI Explorer

ODBC – JDBC - OLEDB

- **ODBC** - Open Database Connectivity
  - Interfaz escrita en C.
  - Propuesta por Microsoft
  - DAO: Modelo de objetos para hacer uso de la tecnología ODBC.
- **JDBC** – Java Database Connectivity
  - Acceso a fuentes de datos desde el lenguaje de programación Java.
- **OLEDB**
  - acceso a cualquier fuente de datos (base de datos, hojas de cálculo,...).
  - Propuesto por Microsoft
  - ADO: Modelo de objetos para hacer uso de esta tecnología.

Administrador de fuentes de datos

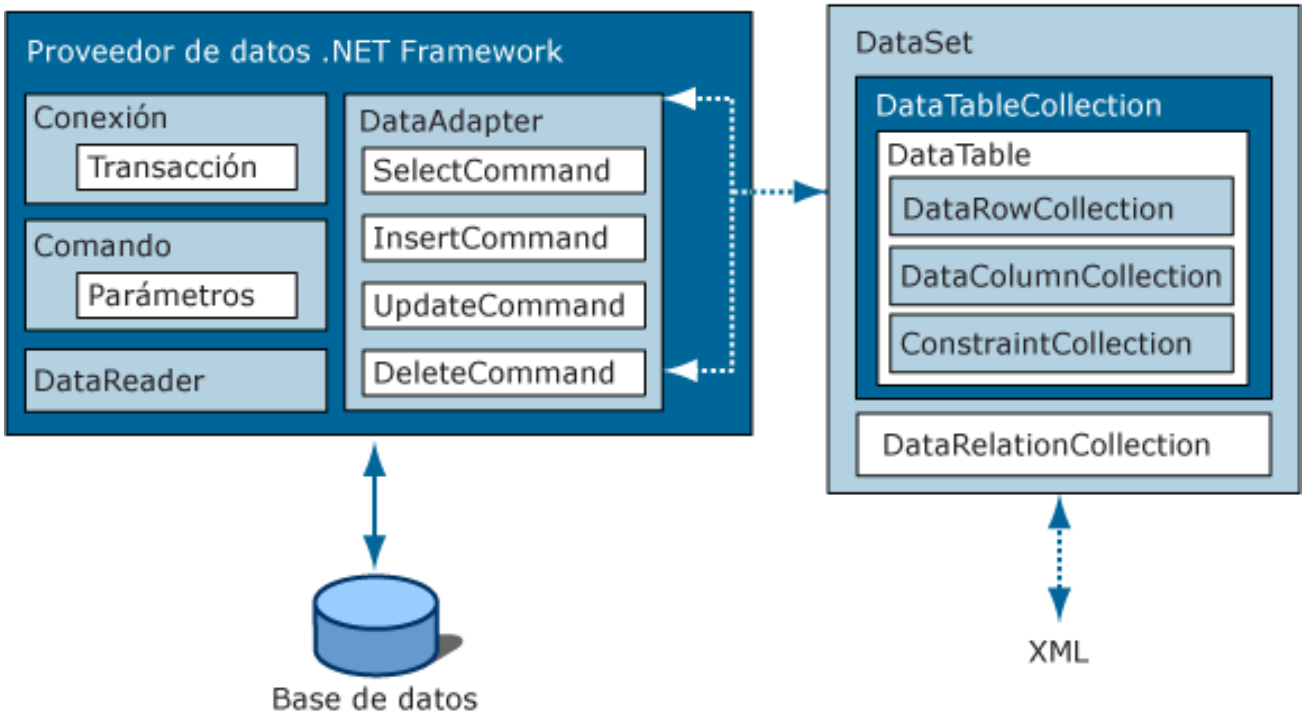


Origen de datos: BD o fichero al que se accede por ODBC

DSN (data source name): nombre del origen de datos

Accediendo a datos con ADO.Net

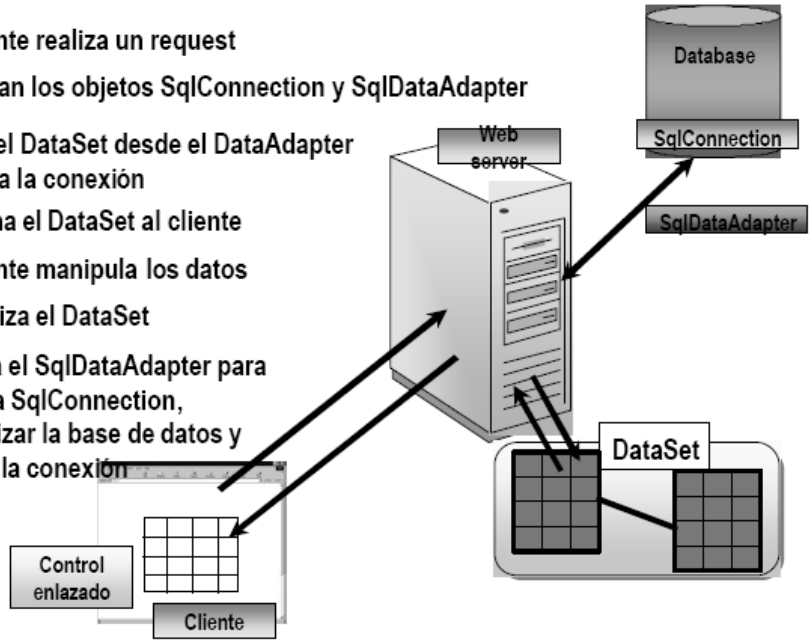
- ADO.NET
  - Componentes incluidos en el Microsoft .NET Framework.
  - Proporcionan soporte para acceder y modificar los datos almacenados en SGBD



http://msdn.microsoft.com/es-es/library/27y4ybxw(v=VS.80).aspx

Ejemplo utilizando ADO (consulta)

1. El cliente realiza un request
2. Se crean los objetos SqlConnection y SqlDataAdapter
3. Llena el DataSet desde el DataAdapter y cierra la conexión
4. Retorna el DataSet al cliente
5. El cliente manipula los datos
6. Actualiza el DataSet
7. Se usa el SqlDataAdapter para abrir la SqlConnection, actualizar la base de datos y cerrar la conexión

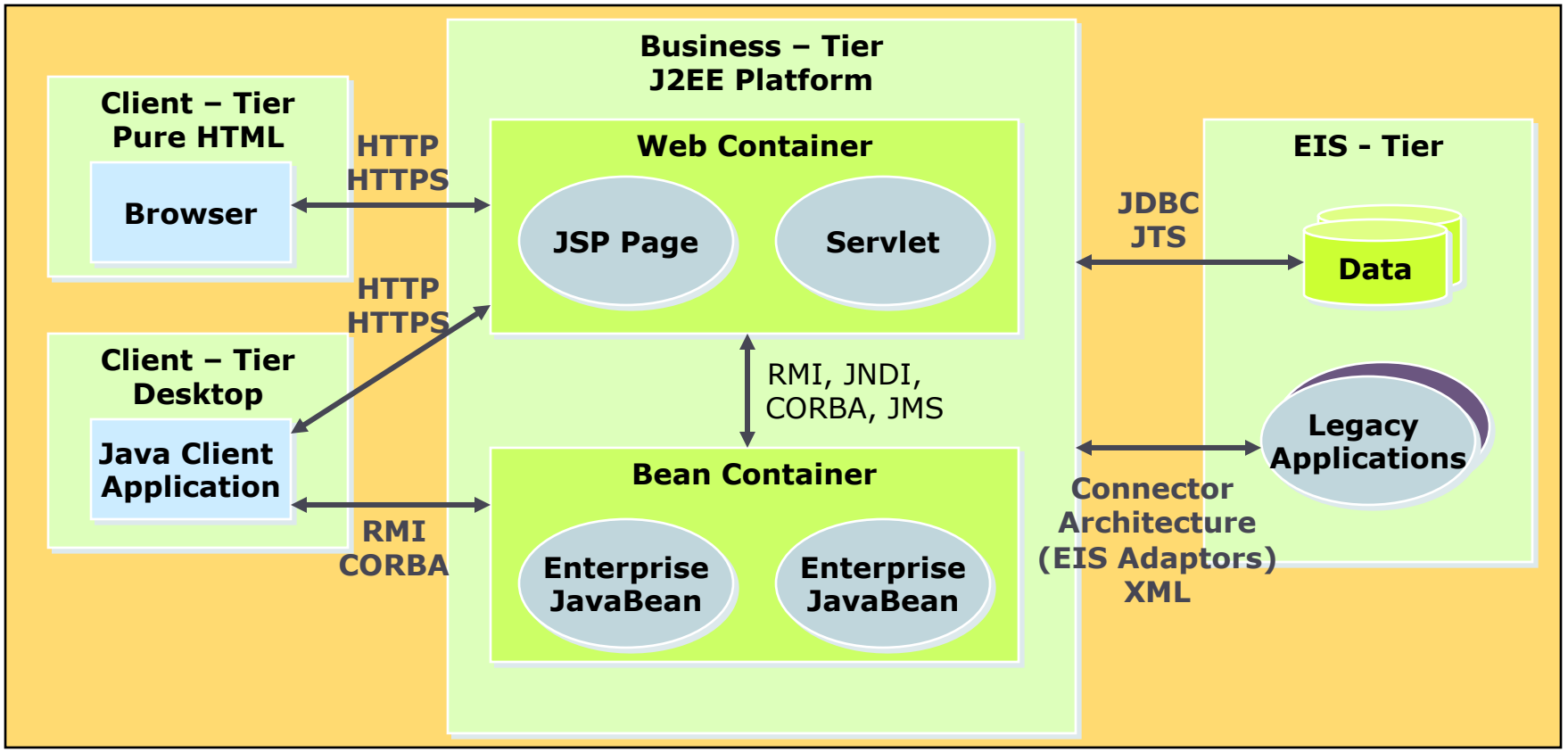


http://msdn.microsoft.com/en-us/library/dw70f090.aspx

```
using System;
using System.Data;
using System.Data.SqlClient;

class Program
{
    static void Main()
    {
        string connectionString = GetConnectionString();
        string queryString =
            "SELECT CategoryID, CategoryName FROM dbo.Categories;
        using (SqlConnection connection =
                   new SqlConnection(connectionString))
        {
            SqlCommand command = connection.CreateCommand();
            command.CommandText = queryString;

            try
            {
                connection.Open();

                SqlDataReader reader = command.ExecuteReader();

                while (reader.Read())
                {
                    Console.WriteLine("\t{0}\t{1}",
                        reader[0], reader[1]);
                }
                reader.Close();
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }

    static private string GetConnectionString()
    {
        // To avoid storing the connection string in your code,
        // you can retrieve it from a configuration file.
        return "Data Source=(local);Initial Catalog=Northwind;"
            + "Integrated Security=SSPI";
    }
}
```

Tecnología J2EE

J2EE define el estándar para desarrollar, implementar y mantener aplicaciones empresariales multicapa

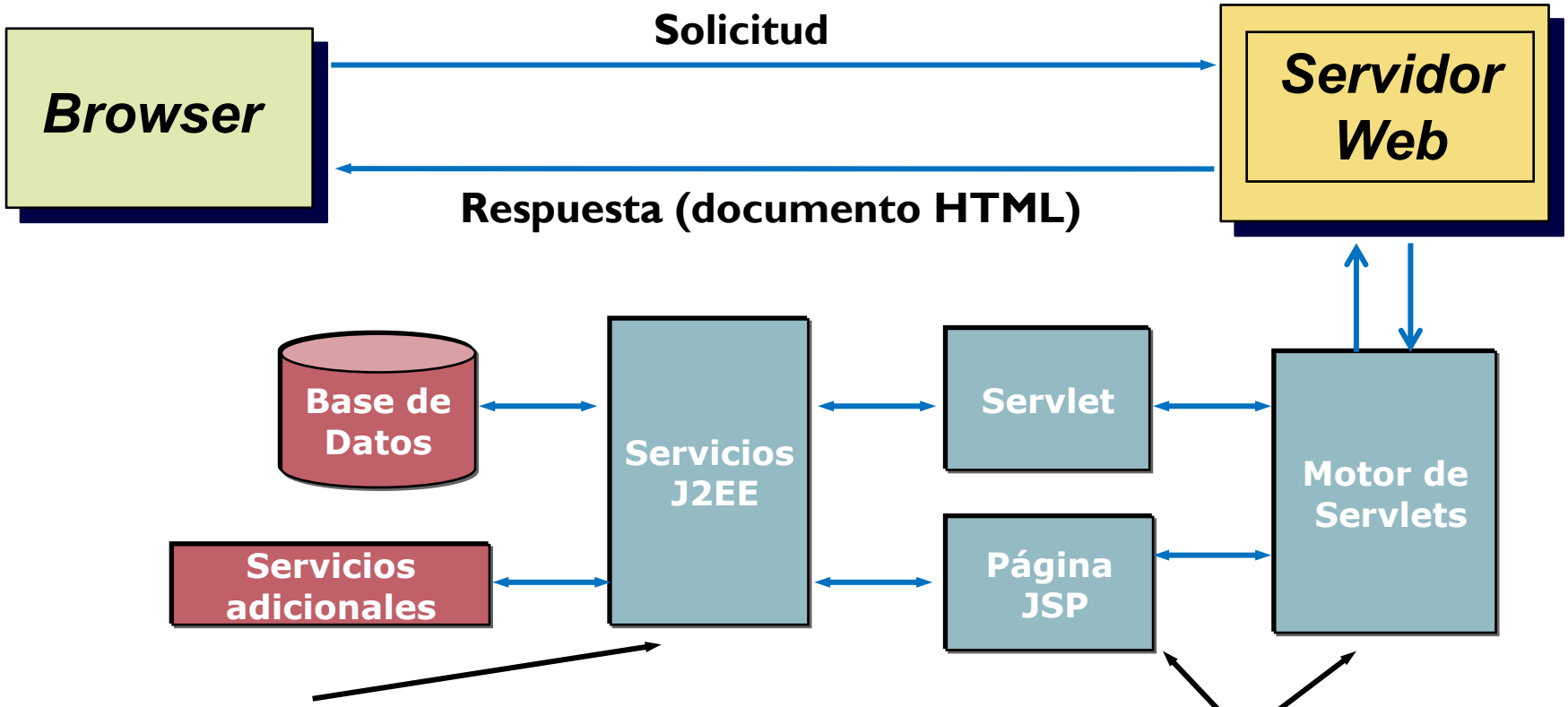


Componentes J2EE

- **Applets**. Componentes de interfaz gráfico, son transportados a través de Internet y ejecutados en el navegador o en otro contenedor de applets.
- **Servlets**. Extensiones del Servidor Web codificados en Java para acceder a sistemas corporativos. Sustituyen a los scripts CGI
- **JSP** (Java Server Pages). Páginas HTML que contienen código Java embebido.

  Servlets y JSP procesan las peticiones Web invocando en caso necesario a funciones del Back-end. El resultado obtenido se convierte dinámicamente en interfaz de cliente (HTML o XML)
- **EJB (Enterprise JavaBeans).**. Permite a múltiples aplicaciones tener acceso de forma concurrente a datos y lógica de negocio. Los EJB se encuentran en un servidor EJB, que no es más que un servidor de objetos distribuidos. Un EJB puede conectarse a cualquier capa, aunque su misión esencial es conectarse con los sistemas de información empresarial (un gestor de base de datos, ERP, etc.)

Tecnología J2EE- Escenario basado en la web más habitual



El escenario más sencillo es usar características de Java (acceso a redes, conexión a BD, etc.) prescindiendo de componentes EJB que se justifican en aplicaciones complejas. Por tanto, la capa web implica recoge la lógica de presentación y la lógica de negocio

Servlets y páginas JSP se instancian por cada proceso al que deben dar respuesta

JDBC

```
public static void JDBCexample(String dbid, String
  userid, String passwd)
     {
     try {
    Class.forName ("oracle.jdbc.driver.OracleDriver");
    Connection conn = DriverManager.getConnection(
     "jdbc:oracle:thin:@aura.bell-labs.com:2000:bankdb",
      userid, passwd);
      Statement stmt = conn.createStatement();
            /*… Do Actual Work ….*/
      stmt.close();
      conn.close();
      }
   catch (SQLException sqle) {
        System.out.println("SQLException : " + sqle);

   }
    }
```

JDBC (continuación)

- Update to database

```
try {
     stmt.executeUpdate(  "insert into account values
                          ('A-9732', 'Perryridge', 1200)");
} catch (SQLException sqle) {
     System.out.println("Could not insert tuple. " + sqle);
}
```

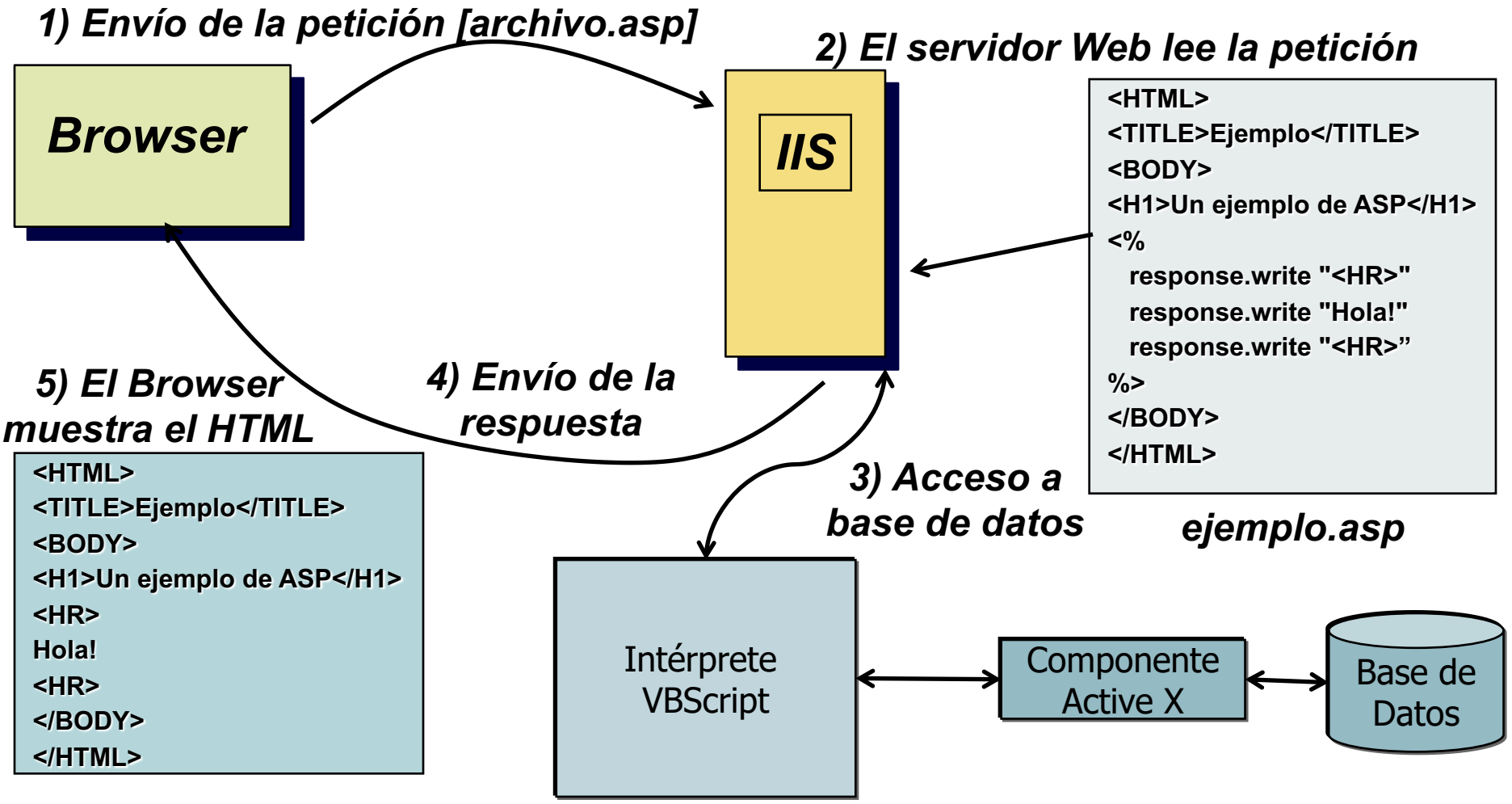
- Execute query and fetch and print results

```
ResultSet rset =
       stmt.executeQuery("select branch_name, avg(balance)
                          from account group by branch_name");
while (rset.next()) {
   System.out.println(
      rset.getString("branch_name") + "  " + rset.getFloat(2));
}
```

Aplicaciones Web con lenguaje de script

- PHP / ASP / JSP
  - Una petición de acceso a un URL hace que el servidor web ejecute un script del servidor
    - Este script manda información de retorno al servidor web
    - El servidor web manda esta información al cliente
  - Los script de servidor son:
    - Más sencillos que los Java servlets o ISAPI
    - Muy populares
  - JSP se convierte en servlet en la primera petición.

Funcionamiento ASP



Ejemplo ASP

```
<%SQLtxt = "SELECT Producto, Cantidad, Precio FROM articulos
set rs = CreateObject("ADODB.Recordset")
rs.Open SQLtxt,"DSN=Mibase"%>
<table>
<%
Do While NOT rs.EOF%>

<tr>
<td><%= rs("Producto")%></td>
<td><%= rs("Cantidad")%></td>
<td align="right"><%= FormatCurrency(rs("Precio"))%></td>
</tr>

<% rs.MoveNext
Loop
rs.Close
</table>
%>
```

ASP vs JSP

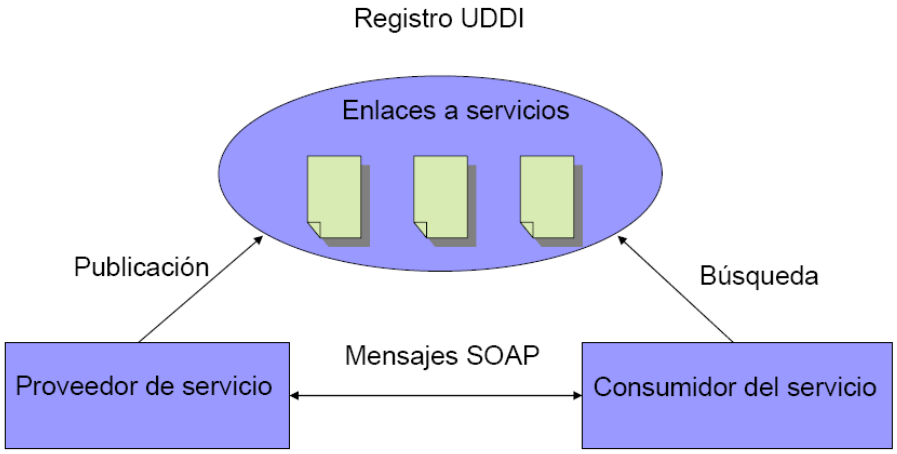
Característica	**ASP**	**JSP**
Servidores	Sólo IIS	Apache, Netscape, IIS ...
Plataformas	Windows	Solaris,Windows,Linux...
Componentes reusables	Componentes COM	Componentes Bean
Seguridad contra fallos de sistema	Basada en seguridad de NT	Seguridad inherente a Java
Lenguajes de programación admitidos	VBScript, JScript, Perl	Java, JavaScript
Posibilidad de integrar orígenes de datos	ODBC, OLEDB	ODBC, JDBC

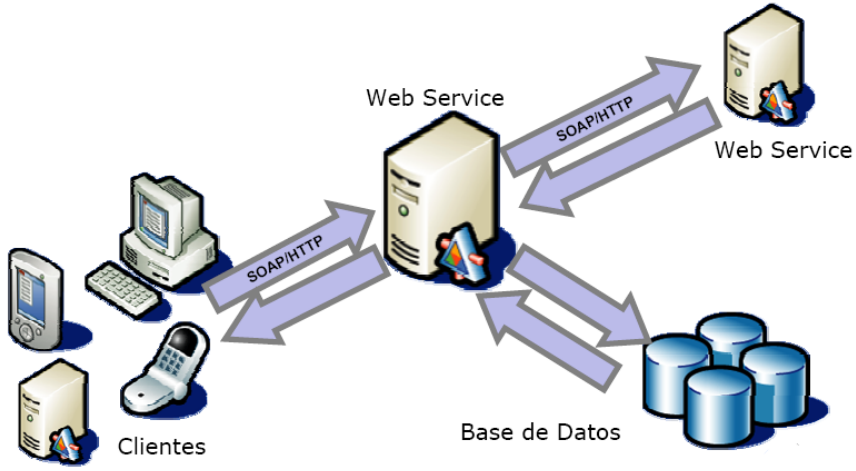
PHP

- PHP - Professional Home Page. 1994 Rasmus Lerdorf
- Puede ejecutarse como módulo de apache o como ***CGI***.
- Junto con apache y MySQL es una opción muy utilizada.
- Apuntes del lenguaje en archivo pdf independiente

Servicios Web

- Componentes que ejecutan procesos o funciones de negocio significativas con una interfaz definida (WSDL) y accesible desde Internet (UDDI). La interacción se realiza mediante intercambio de mensajes con protocolos SOAP y HTTP y el intercambio de información en formato XML.





Servicios Web: Ejemplo de petición al servidor

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

< ?xml version="1.0"?>
< soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-
encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
        <m:GetStockPrice>
                <m:StockName>IBM</m:StockName>
        </m:GetStockPrice>
</soap:Body>

< /soap:Envelope>
```

Servicios Web: Ejemplo de respuesta al cliente

```
HTTP/1.1 200 OK
 Content-Type: application/soap+xml; charset=utf-8
 Content-Length: nnn

 <?xml version="1.0"?>
 <soap:Envelope
 xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
 soap:encodingStyle="http://www.w3.org/2001/12/soap-
encoding">

 <soap:Body xmlns:m="http://www.example.org/stock">
   <m:GetStockPriceResponse>
     <m:Price>34.5</m:Price>
   </m:GetStockPriceResponse>
 </soap:Body>

 </soap:Envelope>
```