Escuela de Sistemas y Tecnologías

Transparencias de **ANALISTA DE SISTEMAS**

Edición 2017 – Materia: **Diseño de Aplicaciones Web**

TEMA: Repaso

Plantel y Contactos

- **Bedelía:**
  - Mail: bedeliasistemas@bios.edu.uy

- **Encargado de Sucursal:**
  - Pablo Castaño
  - Mail: pablocasta@bios.edu.uy

Recursos

- **Recursos Imprescindibles:**
  - Sitio Web de material
    (comunicarse con Bedelía por usuario/contraseña).
  - Transparencias del Curso.
  - Contar con el software necesario

Consideraciones

- Estas transparencias no tienen el objetivo de suplir las clases.
- Por tanto, serán complementadas con ejemplos, códigos, profundizaciones y comentarios por parte del docente.
- El orden de dictado de estos temas está sujeto a la consideración del docente.

Agenda

- POO
- Repaso UML
- MER
- ASP .Net
- ADO .Net
- Arquitectura en Capas
- Ejercicios

BIOS

POO

¿Que es un *Objeto*?

- ***Definición Formal (Rumbaugh):*** "Un objeto es un concepto, abstracción o cosa con un significado y límites claros en el problema en cuestión".

- ***Un objeto posee (Booch):***
  - **Estado**: normalmente cambia en el transcurso del tiempo. Es implementado por un conjunto de propiedades (***atributos***), además de las conexiones que puede tener con otros objetos
  - **Comportamiento**: es modelado por un conjunto de mensajes a los que el objeto puede responder (operaciones que puede realizar)
  - **Identidad**: cada objeto tiene una identidad única, incluso si su estado es idéntico al de otro objeto

¿Que es una *Clase*?

- Una clase es una descripción de un grupo de objetos
- Una clase es una abstracción.
- Posee:
  - Propiedades en común (atributos)
  - Comportamiento similar (operaciones)
  - La misma forma de relacionarse con otros objetos (relaciones)
- Un objeto es una instancia de una clase

Elementos de una Clase

- ***Atributos*** (tipo de acceso privado)
- ***Propiedades*** (permiten manejar los atributos – pueden contener código defensivo)
- ***Constructores*** (hay 3 tipos)
- ***Comportamiento*** (operaciones con o sin métodos asignados)

Relaciones

- Todo sistema abarca muchas clases y objetos.
- Los objetos contribuyen en el comportamiento de un sistema colaborando entre si. La colaboración se logra a través de las relaciones.
- La multiplicidad es el número de instancias que participan en una asociación.
- Las relaciones se programan en función de la multiplicidad del lado de la clase que absorbe la relación :
  - si la multiplicidad es uno se genera un atributo de tipo objeto relacionado
  - si la multiplicidad es muchos (*) se genera un atributo lista de tipo objeto relacionado

Herencia

- Indica que una ***Clase Derivada*** hereda los métodos, operaciones y atributos especificados en su ***Clase Base***.
- La ***Clase Derivada*** además de poseer sus propios métodos, operaciones y atributos, poseerá las características y atributos de la ***Clase Base***.
- Su principal utilidad es la reutilización de código.

Tipos de Elementos

- ***Atributo / Propiedad / Operación de Instancia***: son los atributos u operaciones que solo pueden accederse a través de una instancia (objeto creado) de la clase. Valor único por instancia
- ***Atributo / Propiedad / Operación de clase***: son los atributos u operaciones que solo pueden accederse a través de la propia clase. Valor compartido por todas las instancias existentes de la clase.
- ***Operación Abstracta***: es una operación sin método. Una operación puede ser abstracta en una clase, pero concreta en otra.
- ***Clase abstracta***: es una clase de la cual no se pueden obtener instancias directas. Si una clase contiene al menos una operación abstracta, entonces la clase debe ser abstracta.

Modificadores de Acceso

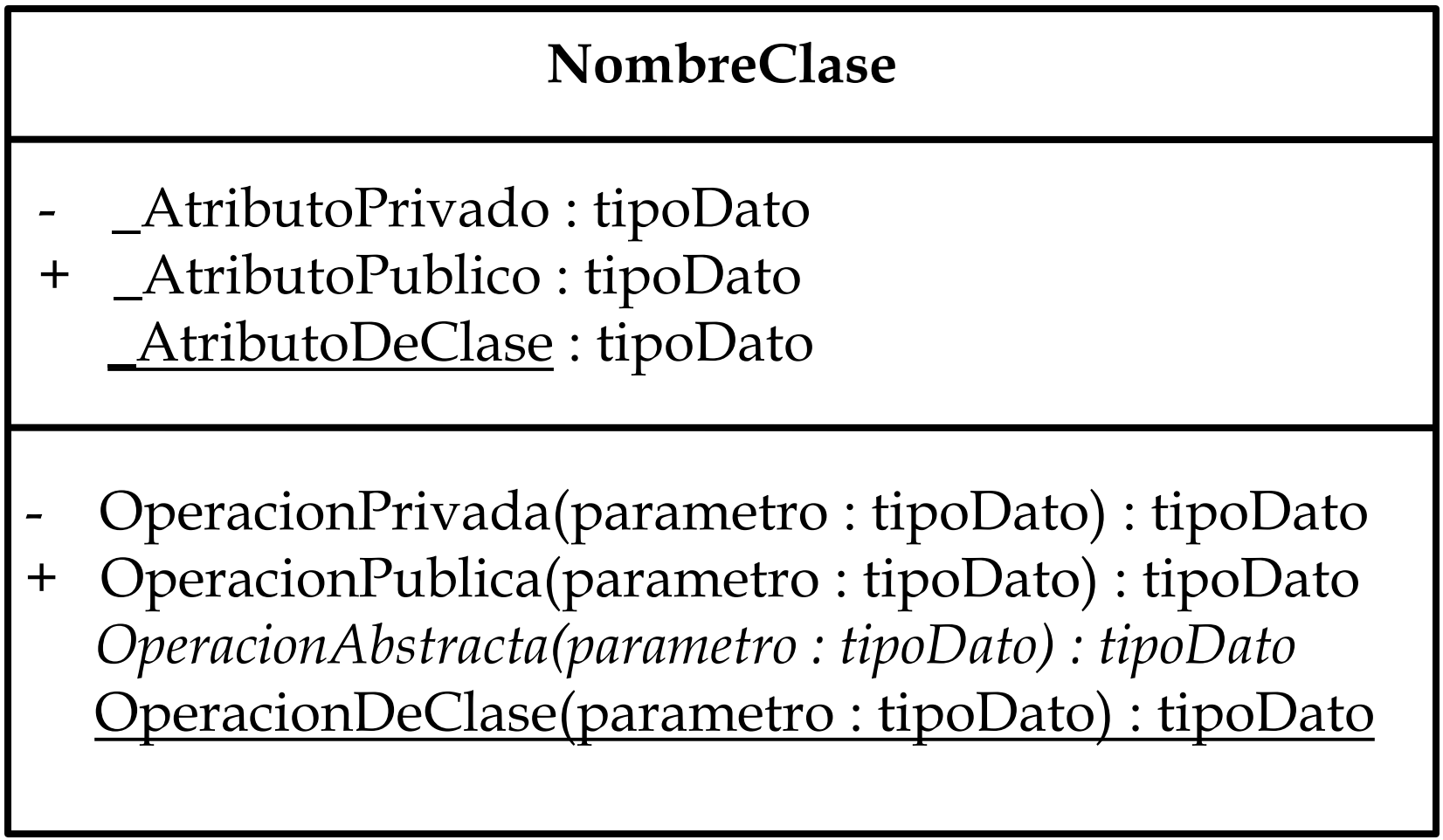
- ***Public***: Cualquier clase puede "ver" los miembros públicos de otra clase
- ***Private***: Sólo la clase puede ver sus propios miembros privados
- ***Protected***: Solo puede ser accedido desde la clase en la que está declarado y desde las clases que hereden de ella
- ***Internal***: Sólo puede ser accedido desde las clases que estén en el mismo paquete lógico que la clase en la que está definido. Ejemplo de paquete: biblioteca de clases, aplicación escritorio.

Otros Conceptos

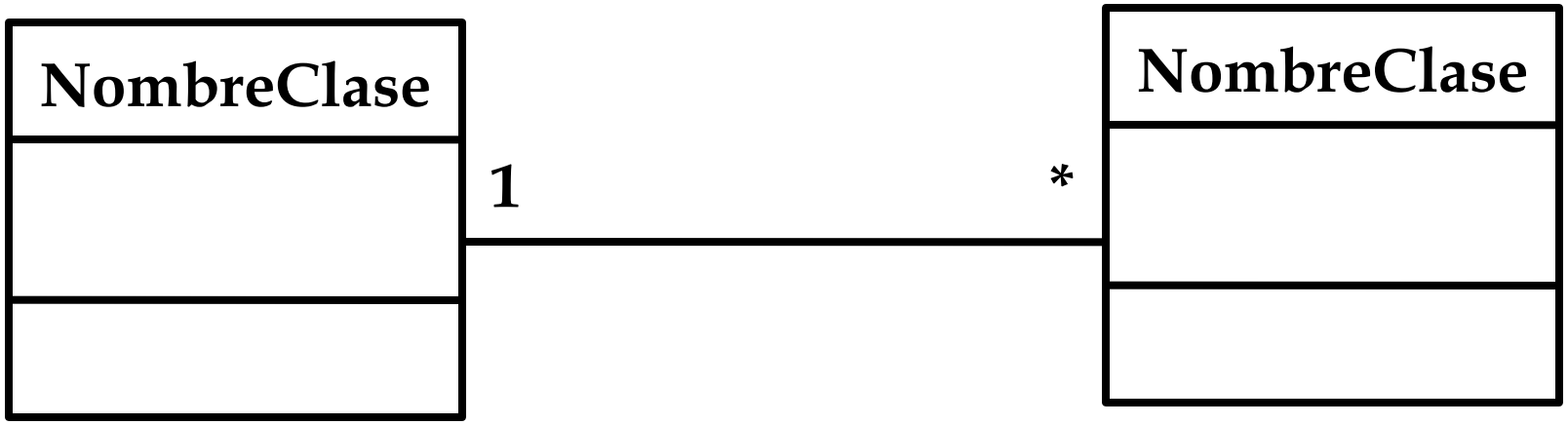
- ***Redefinición de Operaciones***: Una operación redefine a otra operación de su clase base cuando le asigna otro método en la clase derivada
- ***Sobrecarga de Operaciones***: es la capacidad de permitir que muchas operaciones reciban el mismo nombre pero con diferente firma (diferente cantidad y/o tipo de parámetros). Los constructores son un buen ejemplo de sobrecarga.
- ***Casteo***: El casteo permite redefinir el tipo de una referencia. Se hace necesario debido a la reemplazabilidad.

Repaso UML

UML - Clase

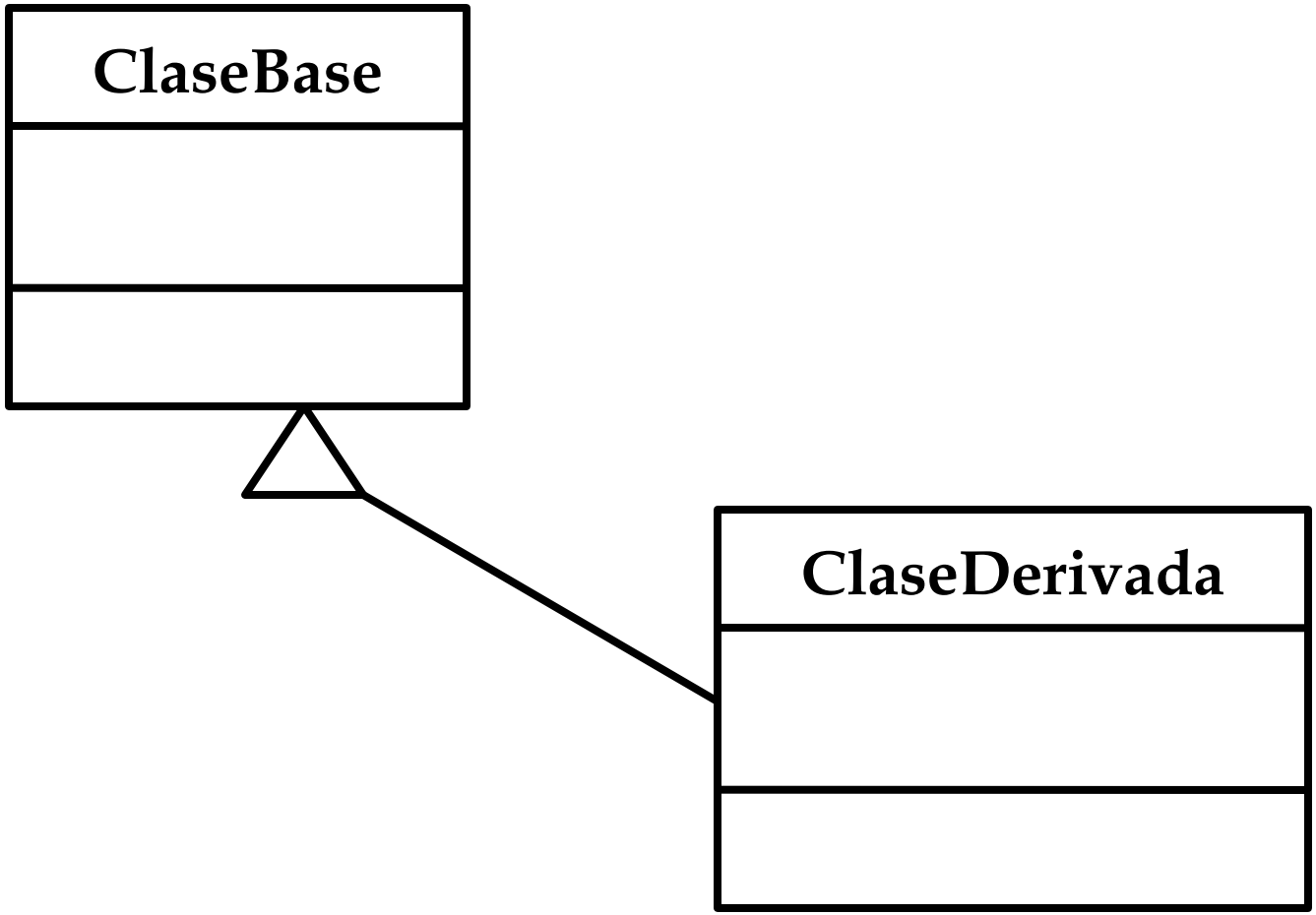

NombreClase
- _AtributoPrivado : tipoDato
+ _AtributoPublico : tipoDato
_AtributoDeClase : tipoDato
- OperacionPrivada(parametro : tipoDato) : tipoDato
+ OperacionPublica(parametro : tipoDato) : tipoDato
OperacionAbstracta(parametro : tipoDato) : tipoDato
OperacionDeClase(parametro : tipoDato) : tipoDato

UML – Asociación



Multiplicidad - Relaciones de tipo Asociación: especifica cuantas instancias de una clase se pueden relacionar a una sola instancia de otra clase.

UML – Herencia



Diagramas

- ***Modelo conceptual***: representa todas los conceptos de la realidad a diseñar. Incluye todas sus relaciones (de tipo asociación, herencia, etc). En esta etapa solo se toman en cuenta los datos que describen cada concepto, operaciones especificas de cada uno y sus relaciones (no implementadas aun en las clases correspondientes).
- ***Diagrama de Clases***: Es un diagrama similar al anterior, pero incluye la programación de las relaciones (las cuales ya no se representan gráficamente), y todos los elementos extras que genere el lenguaje de programación que se utilizo para implementar el modelo conceptual.
- ***Diagrama de Clases de la Arquitectura en Capas***: representa a todas las clases que forman parte de las capas del sistema. Incluye no solo las relaciones entre ellas, sino también la especificación de colaboración (uso) que exista.

MER

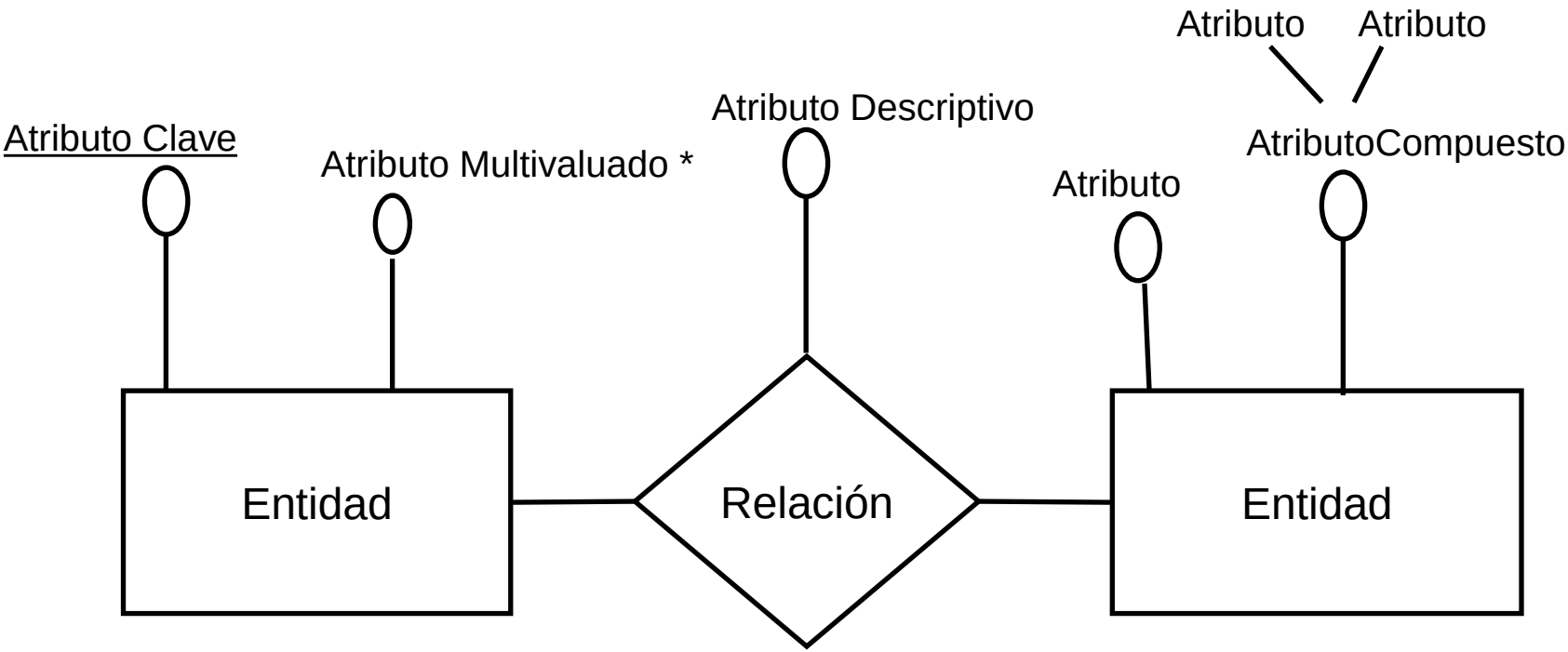
Definiciones (1)

- ***Conjunto de entidades:*** concepto de la realidad que queremos representar, el cual se describe a través de su nombre y sus atributos. Una ***entidad*** es un elemento en el mundo real que es distinguible de todos los demás elementos. Los valores para algún conjunto de atributos pueden identificar una entidad en su conjunto. A este conjunto de atributos se les llama clave.
- ***Conjunto de relaciones***: asociación entre entidades. Puede tener sus propias atributos descriptivos. El ***grado de una relación*** es la cantidad de conjuntos de entidades que vincula.

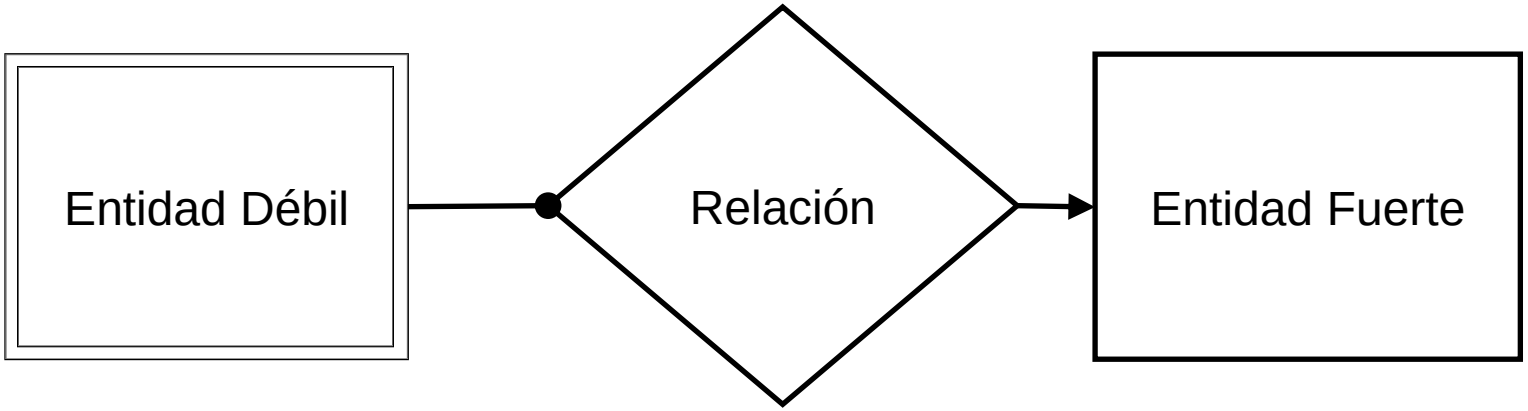
Definiciones (2)

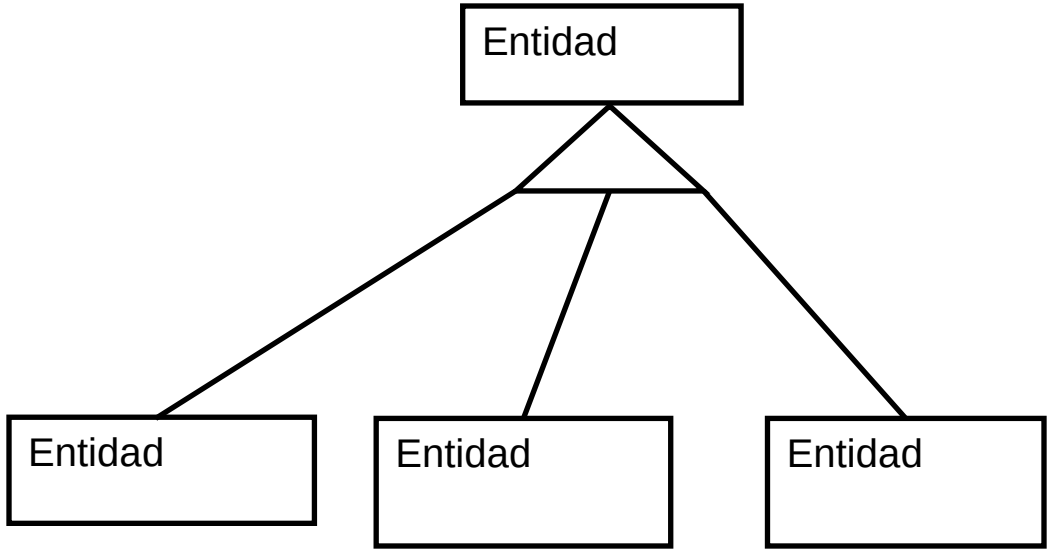
- Una entidad se representa mediante un conjunto de ***atributos***.
  - Cada atributo tiene un conjunto de valores permitidos, llamados el dominio de ese atributo.
  - Clasificación de los atributos según:
    - Estructura (Simples o Compuestos).
    - Cantidad de valores asociados (Atómicos o Multivaluados).
- ***Cardinalidad***: Nos permite indicar cuantas entidades están relacionadas con una entidad de otro conjunto

Representación Grafica (1)



Representación Grafica (2)





ASP .Net

Conceptos Básicos (1)

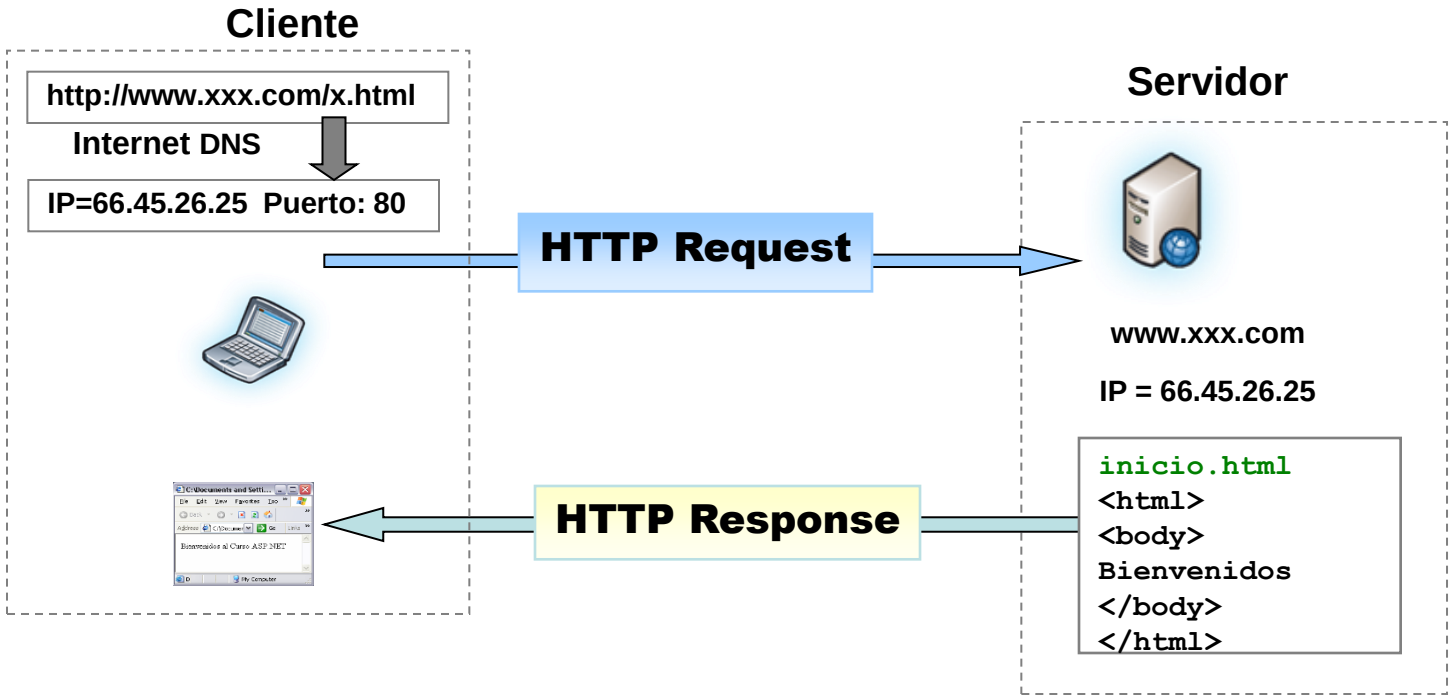
- Aplicación Web (AppWeb)
- ASP.Net
- Servidor Web
- Internet Information Server (IIS)
- Página Web
- Página de contenido Dinámico
- Dirección Virtual

Conceptos Básicos (2)

- Componentes de una AppWeb:
  - ***Partial Class***
    - ***WebForms*** (.aspx) – ***UserControl*** (.ascx)
    - ***Archivos Code-Behind*** (.aspx.cs / .ascx.cs**)**
  - ***Archivos de Configuración*** (formato XML) (Web.config - Machine.config)
  - ***Global.asax***
  - ***Directorio BIN***

Comunicación

- HyperText Transfer Protocol (HTTP)



Ciclo de Vida de un WebForm

1. Se genera un ***HttpRequest*** desde un navegador
2. ***Init***
3. Proceso datos ***Post***
4. ***Load***
5. ***PostBack***
6. ***Render***
7. ***UnLoad***

ASP .Net (1)

- La clase **HttpResponse** encapsula información de una operación de respuesta de página de ASP.NET. El método **Redirect**() permite redireccionar el navegador cliente hacia una nueva dirección que no sea el WebForm actual.
- La sesión de un usuario comienza cuando el navegador cliente peticiona la primer página de una aplicación Web; y termina cuando expira el tiempo de duración, o cuando el navegador cliente no interactúa más con la aplicación Web. Los eventos de comienzo y final de sesión se pueden capturar dentro del archivo **Global.asax**. La colección **Session** permite almacenar información relativa solamente a una sesión de usuario.

ASP .Net (2)

- El estado de una aplicación Web comienza a funcionar en el momento que por primera vez es solicitado un objeto de ésta; y se pierde cuando la aplicación deja de funcionar (se reinicia, o se da de baja el sitio del servidor). Los eventos de comienzo y final de aplicación se pueden capturar por medio del archivo **Global.Asax**. La colección **Application** permite almacenar información visible a todas las sesiones de usuario.

WebForms

- Propiedades Básicas
  - BgColor
  - Title
  - IsPostBack
  - IsValidate
- Evento Básico
  - Page_Load()

Web Controls (1)

- Control **Button** (btn):
  - Propiedades Básicas

    `Text`
  - Evento Básico

    `Click()`
- Control **LinkButton** (lbtn):
  - Propiedades Básicas

    `PostBackUrl`

    `Text`
- Control **TextBox** (txt):
  - Propiedades Básicas

    `Enabled`

    `Text`

Web Controls (2)

- Control **CheckBox** (ckb) - **RadioButton** (rbtn):
  - Propiedades Básicas

    `Checked`

    `Text`
  - Evento Básico

    `CheckedChange()`
- Control **Image** (img):
  - Propiedades Básicas

    `ImageUrl`

    `AlternateText`

Web Controls (3)

- Control **ListBox** (lb) - **DropDownList** (ddl):
  - Propiedades Básicas
    - `SelectedIndex`
    - `SelectedValue`
    - `DataSource`
    - `DataTextField`
    - `DataValueField`
  - Evento Básico
    - `SelectedIndexChanged()`
  - Método Básico
    - `DataBind()`

WebControls (4)

- Control **DataGrid** (dg) :
  - Propiedades Básicas

    `Columns`

    `Caption`

    `DataSource`
  - Método básico:

    `DataBind()`

ADO .Net

SqlConnection

- Es la clase que permite realizar una conexión con la base de datos.
- Constructores:

```
new SqlConnection();
new SqlConnection(connectionString);
```

- Propiedades básicas:

```
.ConnectionString
.ConnectionTimeOut
```

- Métodos básicos:

```
.Open()
.Close()
```

SqlCommand

- Representa una sentencia SQL o un PA.
- Constructores:

```
new SqlCommand();
new SqlCommand("sql/SP", objSqlConnection);
```

- Propiedades básicas:
  - `CommandType`
  - `Parameters`
- Métodos básicos:

```
.ExecuteReader()
.ExecuteNonQuery()
```

SqlParameter

- Representa un parámetro para los objetos ***SqlCommand***
- Constructores:

```
new SqlParameter("nomParam", valorParam);
```

  - Para parámetros de Entrada (**Input**)

```
new SqlParameter("nomParam", tipoDato);
```

  - Para parámetros de Salida (**Output**) y de retorno (**ReturnValue**)

- Propiedad básica:
  - `Direction`
  - `Value`

SqlDataReader

- Representa un flujo de datos entre la aplicación y la base de datos. Los registros se leen de a uno y sólo hacia delante (*read-only, forward-only*). La información es de sólo lectura.
- Métodos básicos:
  - `.Read()`
  - `.HasRows()`
  - `.Close()`

Arquitectura en Capas

Componentes

- La **Arquitectura de Software** es la primera etapa del diseño de software, donde se describe un conjunto de unidades arquitectónicas y de cómo esas unidades se relacionan entre sí para solucionar los problemas.
- Estas "unidades" son generalmente también llamados "componentes" de la arquitectura.
- Es importante notar que un ***componente*** no es una clase, sino un conjunto de clases.

Arquitectura en 3 Capas (1)

- En este tipo de arquitectura se coloca la interfaz de usuario en una capa, la lógica del negocio en otra y el acceso a la base de datos en otra.
- Ventajas:
  1. Se reutiliza código.
  2. Las modificaciones en la capa lógica no siempre se tienen que reflejar como modificaciones en la capa de presentación, ni las modificaciones de la capa de datos se tiene porque reflejar ni en la lógica ni en la presentación (encapsulación).
  3. Muy utilizada en proyectos medianos o grandes o que requieren muchos cambios o actualizaciones.

Arquitectura en 3 Capas (2)

- Desventajas:
  1. Se escribe más código.
  2. El desarrollo del software suele ser más largo y más costoso en un principio, aunque futuras modificaciones, actualizaciones y evoluciones resultarán mucho más sencillas, rápidas y menos costosas.
  3. No es práctico para proyectos chicos o que no se espera que tenga cambios, ni actualizaciones, ni evoluciones.

Arquitectura - Propuesta por Microsoft

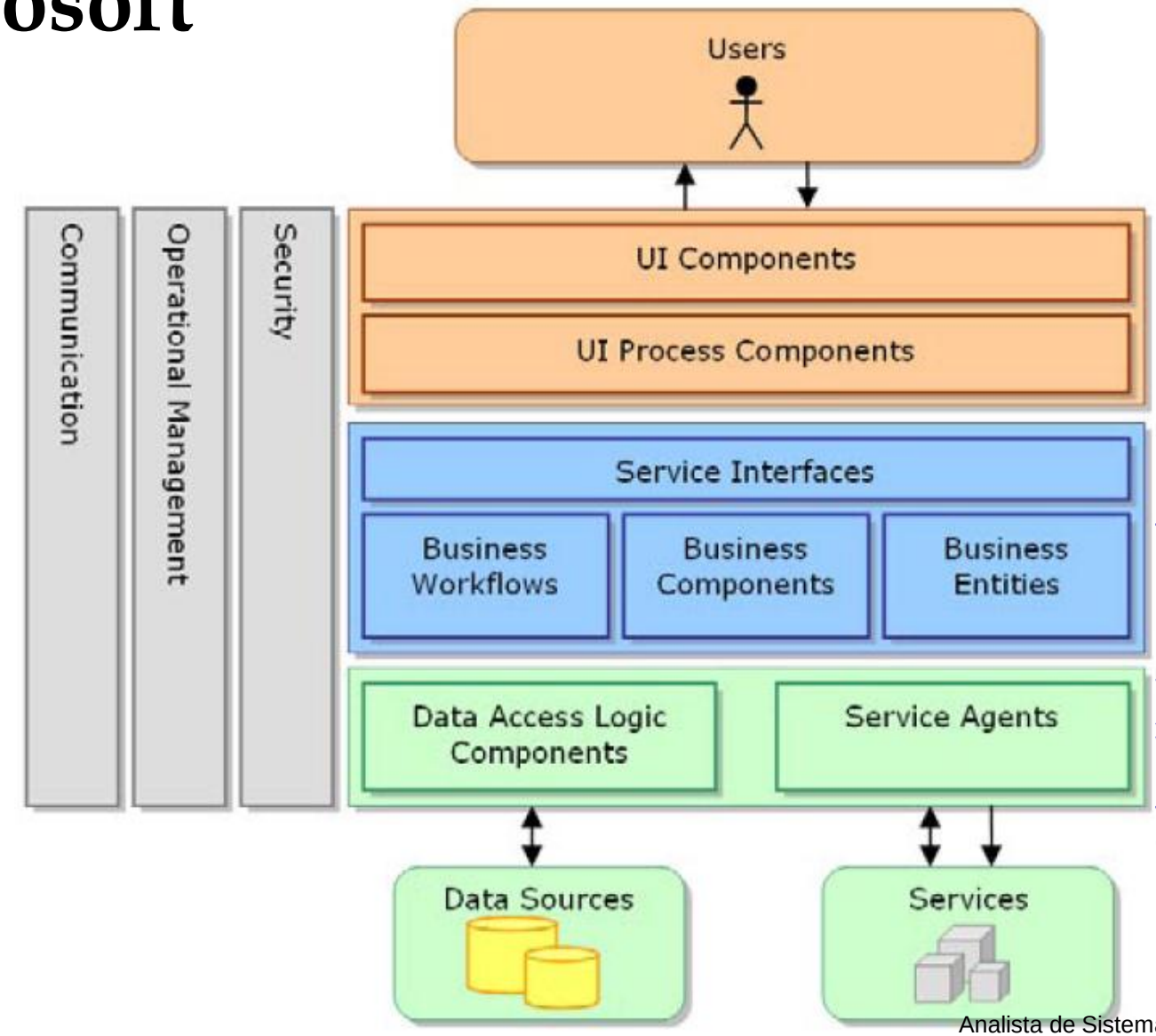


Grafico Publicado por Microsoft

Arquitectura Simplificada

Grafico Publicado por Microsoft

Capa Presentación

- Las principales responsabilidades de sus clases son:
  - Capturar datos del usuario y mostrar los resultados a éste.
  - Contener todo el código relacionado con los formularios así como el diseño gráfico de los mismos.
  - Contener los manejadores de eventos que invocarán las operaciones de la Capa Lógica.

Capa Lógica

- Las principales responsabilidades de sus clases son:
  - Resolver los problemas del negocio para el cual el software fue construido. Es decir, proveer las operaciones del sistema necesarias.
  - Contener clases independientes tanto de la presentación que se utilice así como de la forma de persistir los datos.
  - Responder a las invocaciones de la Presentación.
  - Colaborar con otras operaciones de la Lógica (así como utilizar las entidades compartidas) y con operaciones de la Persistencia para generar el resultado necesario de cada operación.

Capa Persistencia

- Las principales responsabilidades de sus clases son:
  - Recuperar y persistir información desde y hacia la BD.
  - Contener todo el código ADO.NET necesario.
  - Hacer el mapeo entre objetos y registros (pues la BD no guarda objetos sino registros en tablas).
  - Conocer los nombres de los Stored Procedures de la BD.