Célula Académica
UABC-Live .net

Universidad Autónoma de Baja California
Facultad de Ciencias Químicas e Ingeniería

http://uabc-live-net.spaces.live.com/

Sesión No. 5

Introducción a Microsoft .NET

Expositores:

Almanza Sevilla Tania Elena (**thanya86@hotmail.com)**

Carlos Alberto Cabrera González (**carlos5686@hotmail.com)**

Fecha: 19 de Octubre de 2006

Programa Microsoft Desarrollador Cinco Estrellas

Estrella 1
Introducción a Microsoft .NET

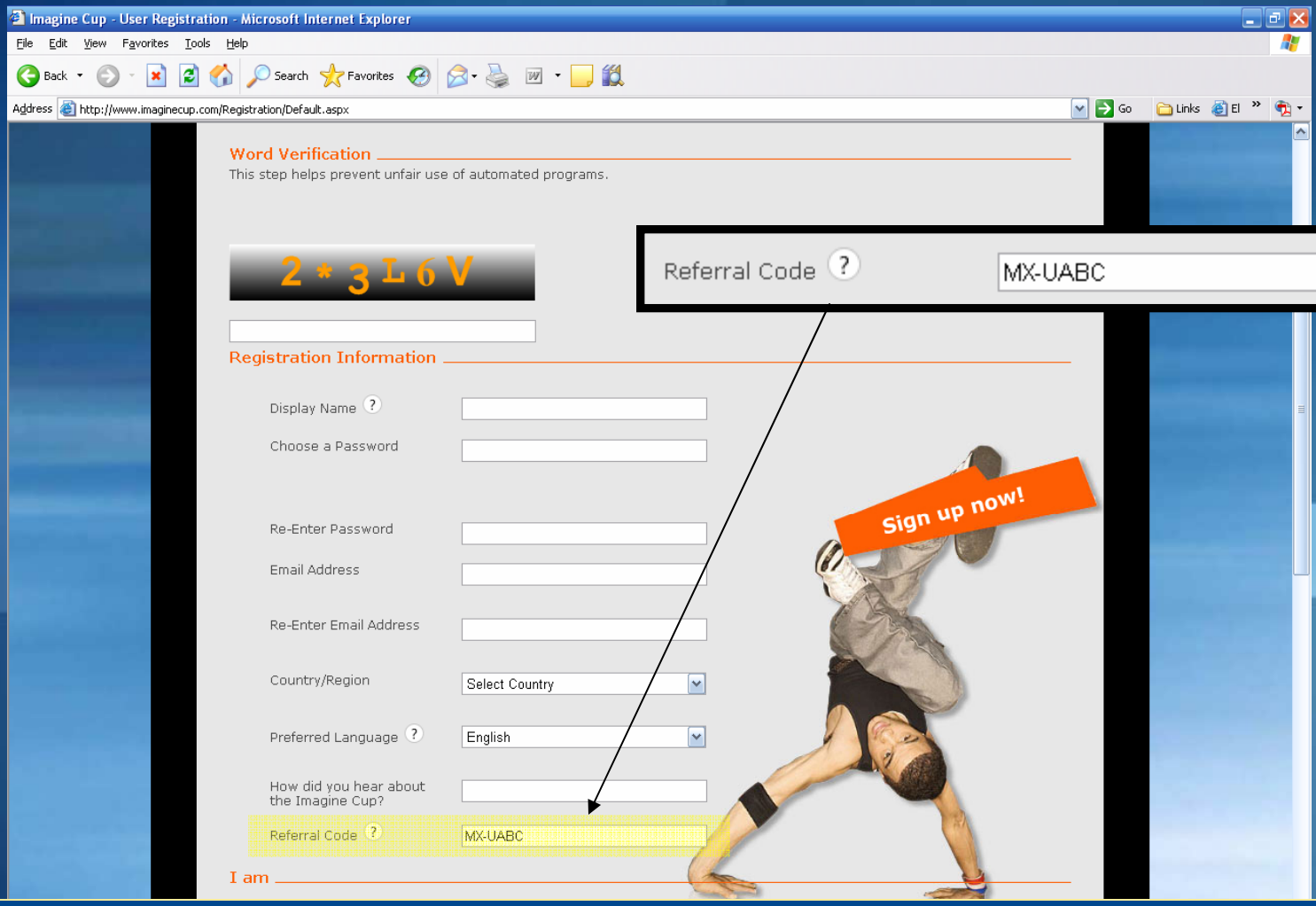
Objetivo

- Presentar una introducción a la plataforma de desarrollo Microsoft .NET, describiendo sus principios básicos de funcionamiento, su arquitectura de componentes y sus principales bibliotecas reutilizables, mostrando además las novedades introducidas en la última versión de la misma.

Prerrequisitos

- Poseer los conocimientos proporcionados por la Estrella 0 del programa
- Haber aprobado el examen correspondiente a la Estrella 0 del programa
- Conocimientos fundamentales de bases de datos relacionales, incluyendo
  - Modelo Relacional
  - Protocolos de Acceso (ODBC/OLEDB)
  - Lenguaje SQL

Imagine Cup 2007



Regístrate utilizando el código MX-UABC en la parte de “Referral Code” para recibir un DVD.

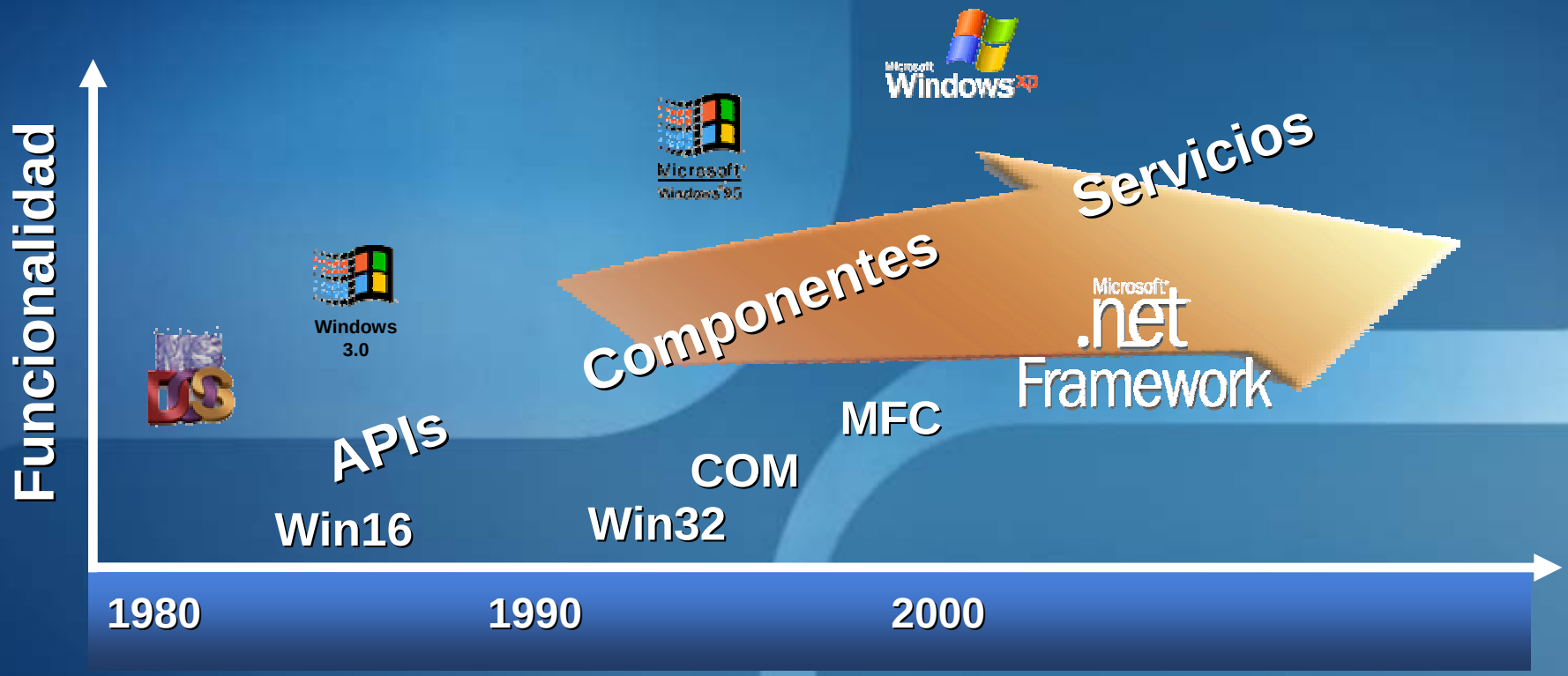
Temas a Tratar

- Introducción a Microsoft .NET
- Componentes Fundamentales
- Funcionamiento Interno del CLR
- Bibliotecas Principales
- Ventajas de .NET
- Herramientas de Desarrollo .NET
- Novedades en .NET 2.0

Temas a Tratar

- Introducción a Microsoft .NET
  - ¿Qué no es .NET?
  - ¿Qué es .NET?
  - .NET Como evolución de COM

Paradigmas de Programación



¿Qué NO es .NET?

- .NET no es un Sistema Operativo
- .NET no es un Lenguaje de Programación
- .NET no es un Entorno de Desarrollo
- .NET no es un Servidor de Aplicaciones
- .NET no es un producto empaquetado que se pueda comprar como tal

¿Qué es .NET?

- Plataforma de Desarrollo compuesta de
  - Entorno de Ejecución (Runtime)
  - Bibliotecas de Funcionalidad (Class Library)
  - Lenguajes de Programación
  - Compiladores
  - Herramientas de Desarrollo (IDE & Tools)
  - Guías de Arquitectura
- La evolución de la plataforma COM

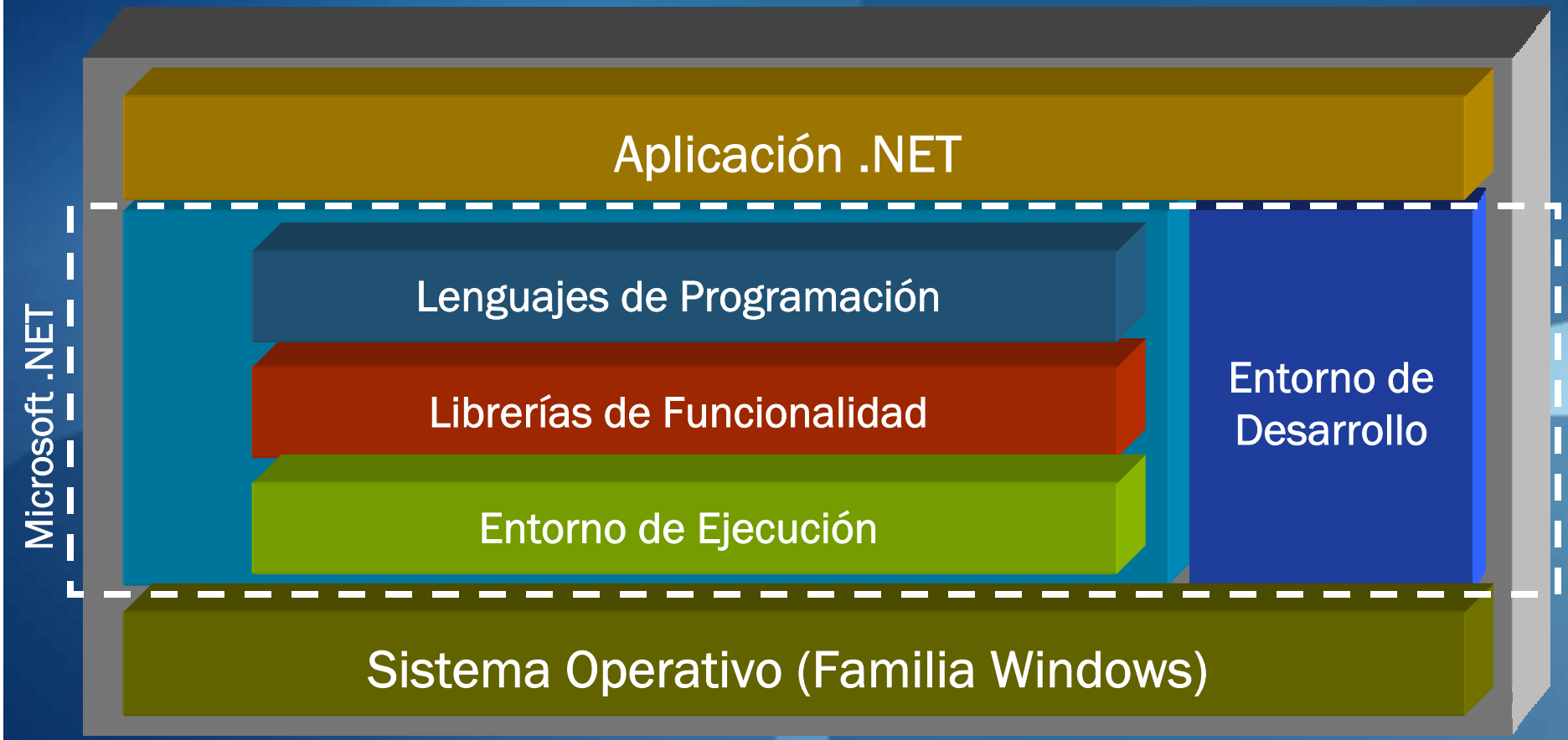
Características de .NET (1/2)

- Plataforma de ejecución intermedia
- 100% Orientada a Objetos
- Multilenguaje
- Plataforma Empresarial de Misión Crítica

Características de .NET (2/2)

- Modelo de Programación único para todo tipo de aplicaciones y dispositivos de hardware
- Se integra fácilmente con aplicaciones existentes desarrolladas en plataformas Microsoft
- Se integra fácilmente con aplicaciones desarrolladas en otras plataformas

Plataforma de Ejecución Intermedia



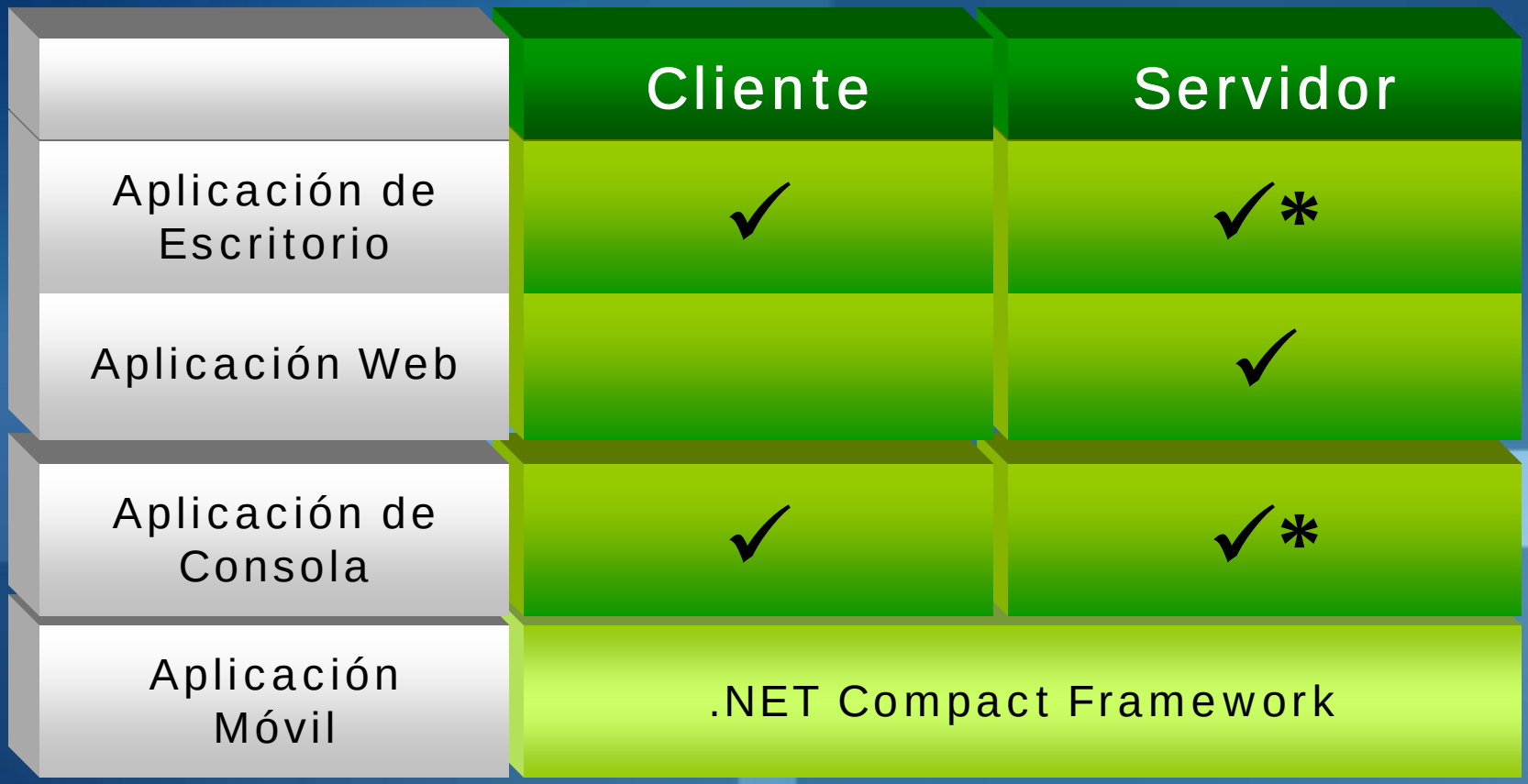
.NET como evolución de COM

- Entorno de Ejecución (Runtime)
  - COM: Windows
  - .NET: Common Language Runtime
- Librerías de Funcionalidad
  - COM: Algunas (ADO, FSO, etc.)
  - .NET: Muy extensa (.NET Framework Class Library)
- Lenguajes de Programación
  - COM: VB, C++, VFP, ASP, J++
  - .NET: Common Language Specification
- Entorno de Desarrollo (IDE)
  - COM: Uno para cada lenguaje
  - .NET: Uno independiente del lenguaje (VS.NET)

¿Qué es el .NET Framework?

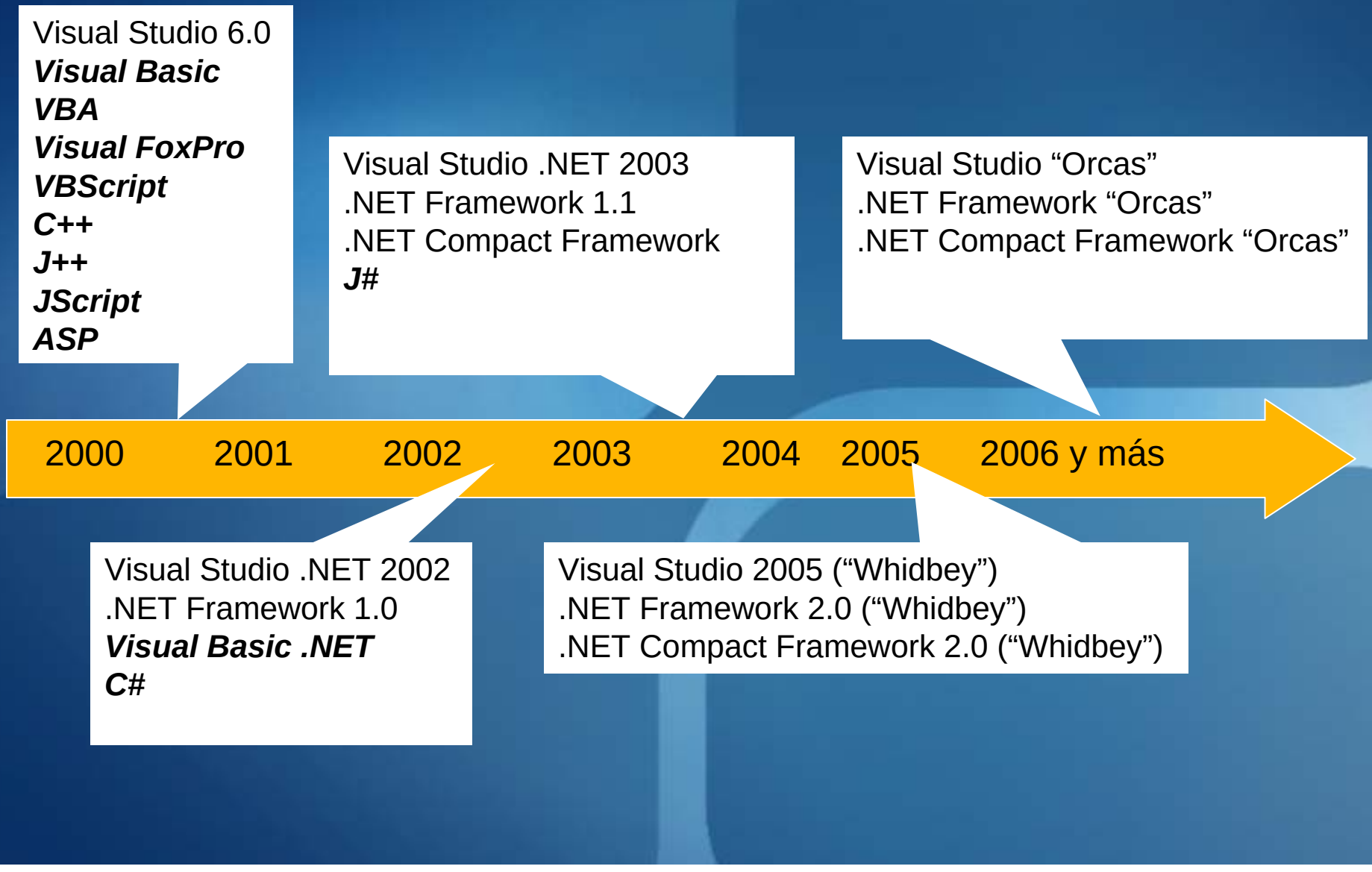
- Paquete de software fundamental de la plataforma .NET. Incluye:
  - Entorno de Ejecución (Runtime)
  - Bibliotecas de Funcionalidad (Class Library)
- Se distribuye en forma libre y gratuita
- Existen tres variantes principales:
  - .NET Framework Redistributable Package
  - .NET Framework SDK
  - .NET Compact Framework
- Está instalado por defecto en Windows 2003 Server o superior

¿Dónde instalar el .NET Framework?



* Sólo si la aplicación es distribuída

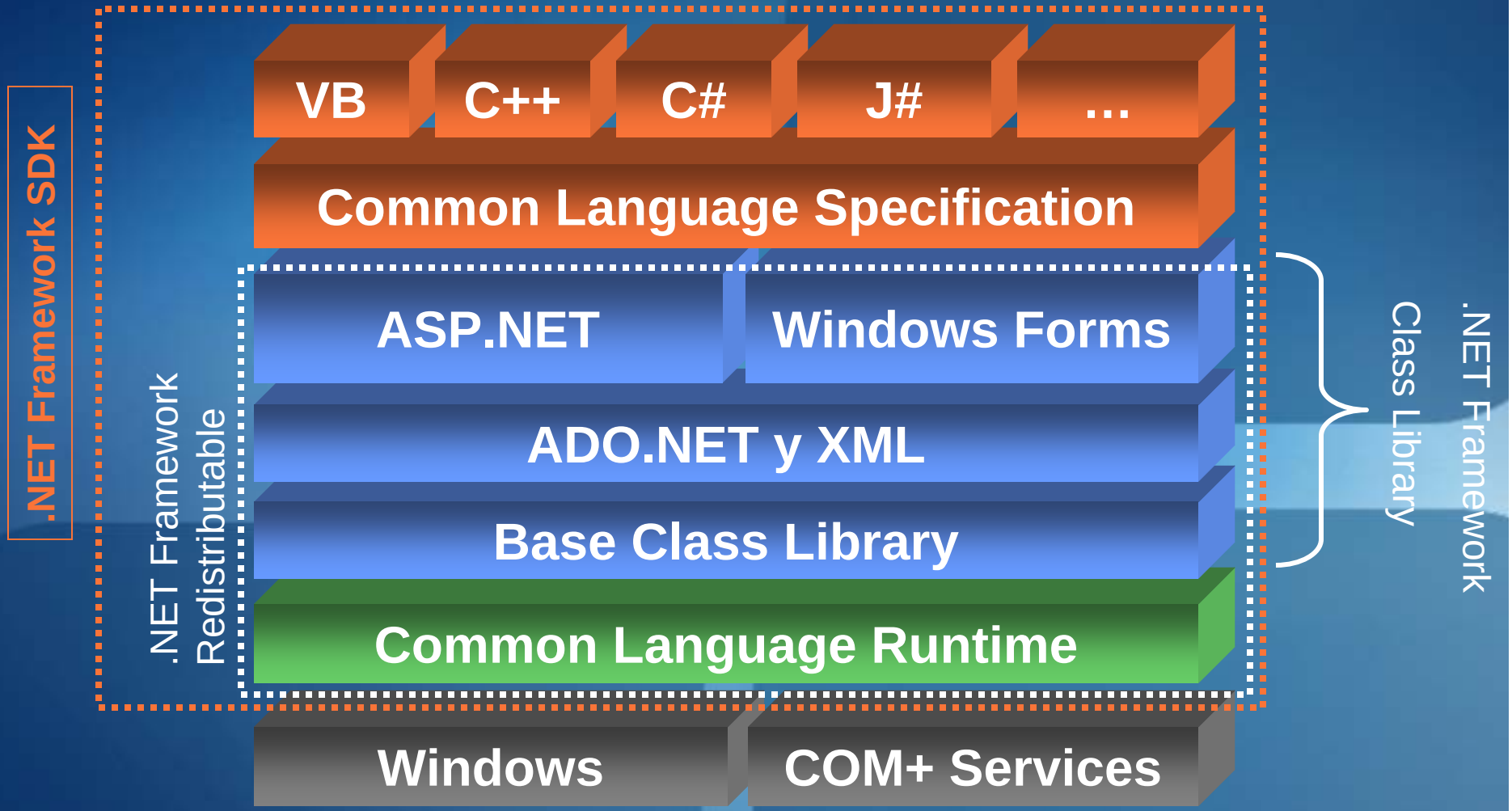
Línea del tiempo de .NET



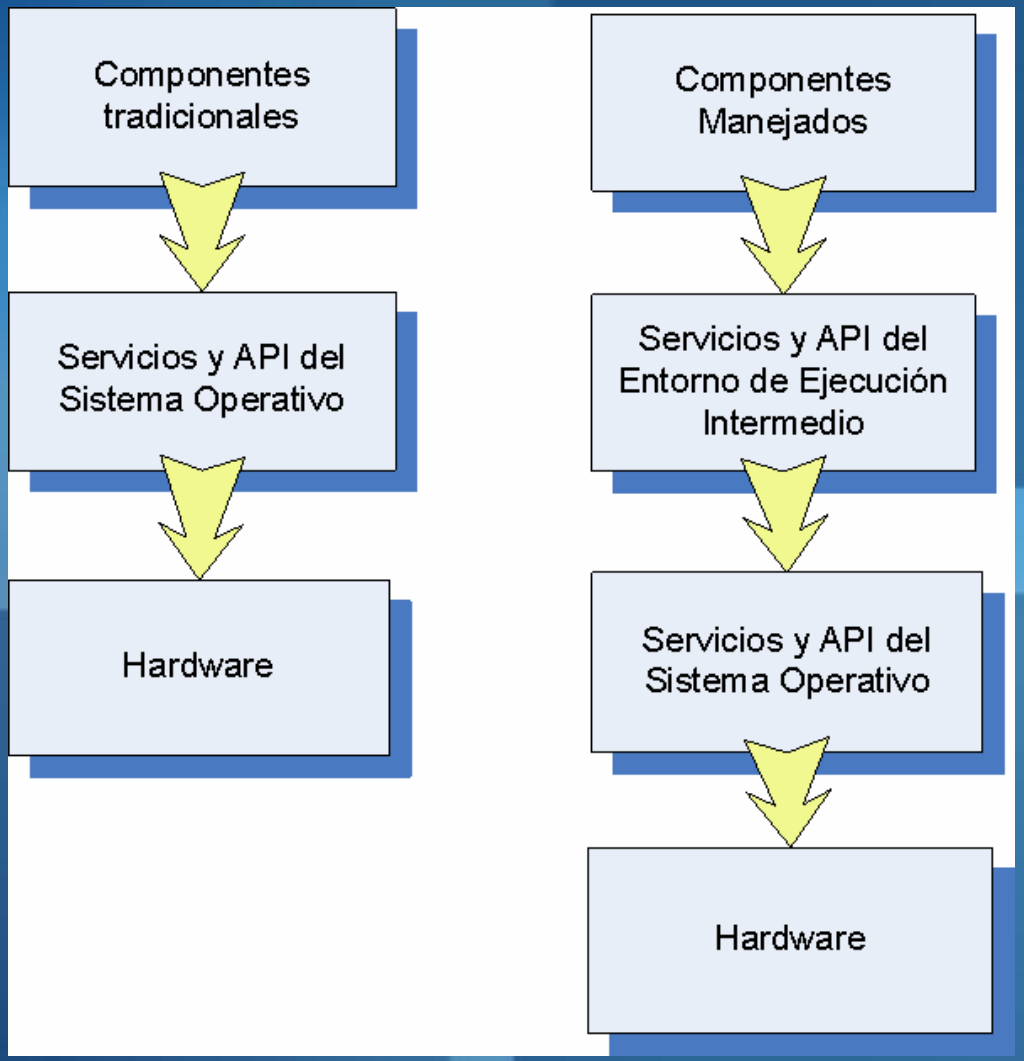
Temas a Tratar

- Introducción a Microsoft .NET
- Componentes Fundamentales
  - Arquitectura
  - Common Language Runtime (CLR)
  - Microsoft Intermediate Language
  - Assemblies
  - .NET Class Library
  - Common Language Specification (CLS)

Arquitectura del .NET Framework



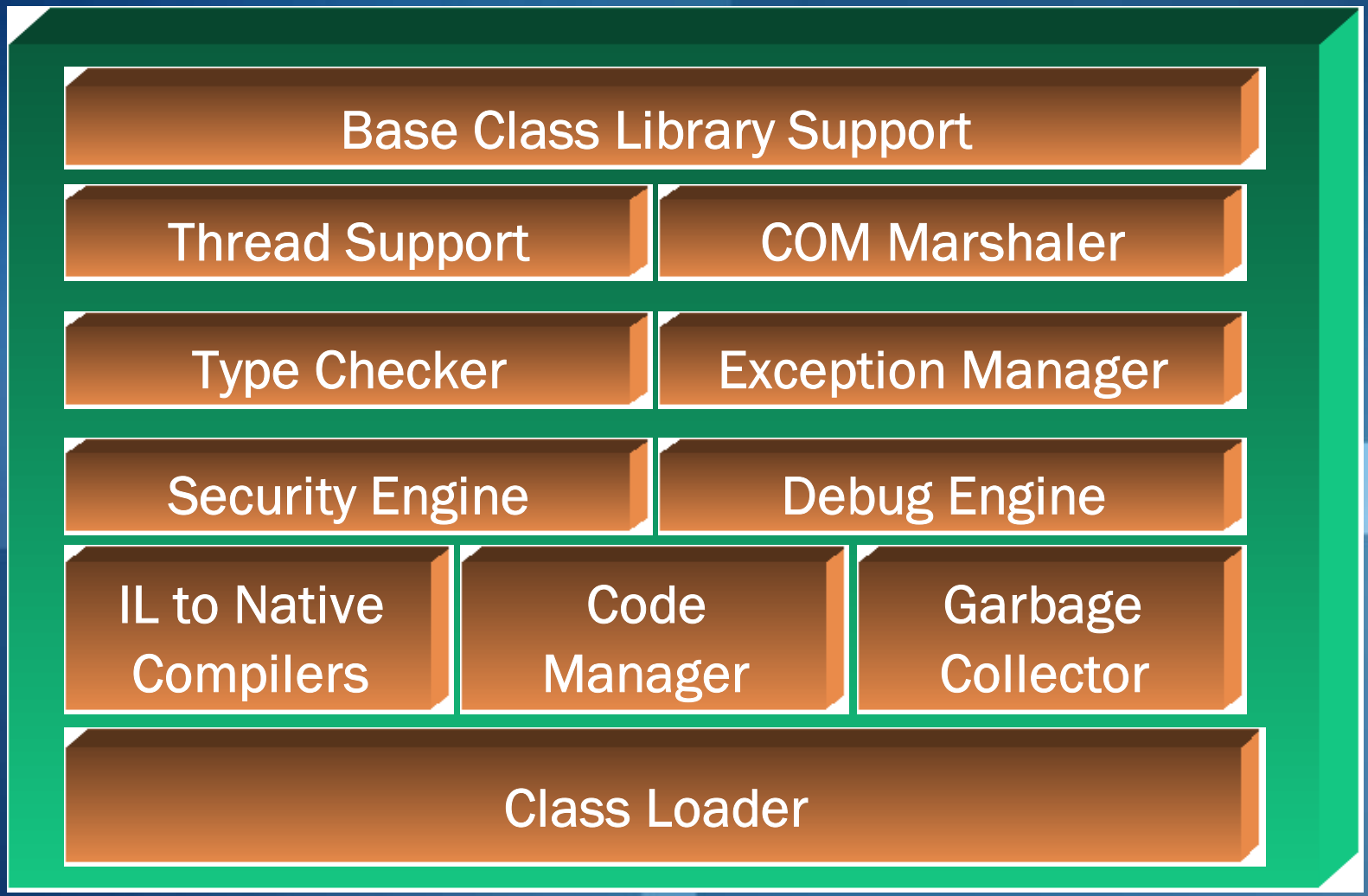
CLR - Arquitecturas de Ejecución de Aplicaciones



CLR – Common Language Runtime

- El CLR es el motor de ejecución (runtime) de .NET
- Características
  - Compilación Just-In-Time (JIT)
  - Gestión automática de memoria (Garbage Collector)
  - Gestión de errores consistente (Excepciones)
  - Ejecución basada en componentes (Assemblies)
  - Gestión de Seguridad
  - Multithreading

CLR – Componentes Internos



CLR – Proceso de Compilación

CLR - MSIL

```
.method private hidebysig static void Main(string[] args) cil
        managed {
.entrypoint
maxstack 8
L_0000: ldstr "Hola Mundo"
L_0005: call void [mscorlib]System.Console::WriteLine(string)
L_000a: ret
}
```

¿Qué es un “Assembly”?

- Un Assembly es la unidad mínima de ejecución, distribución, instalación y versionado de aplicaciones .NET

Assemblies - Aplicaciones .NET

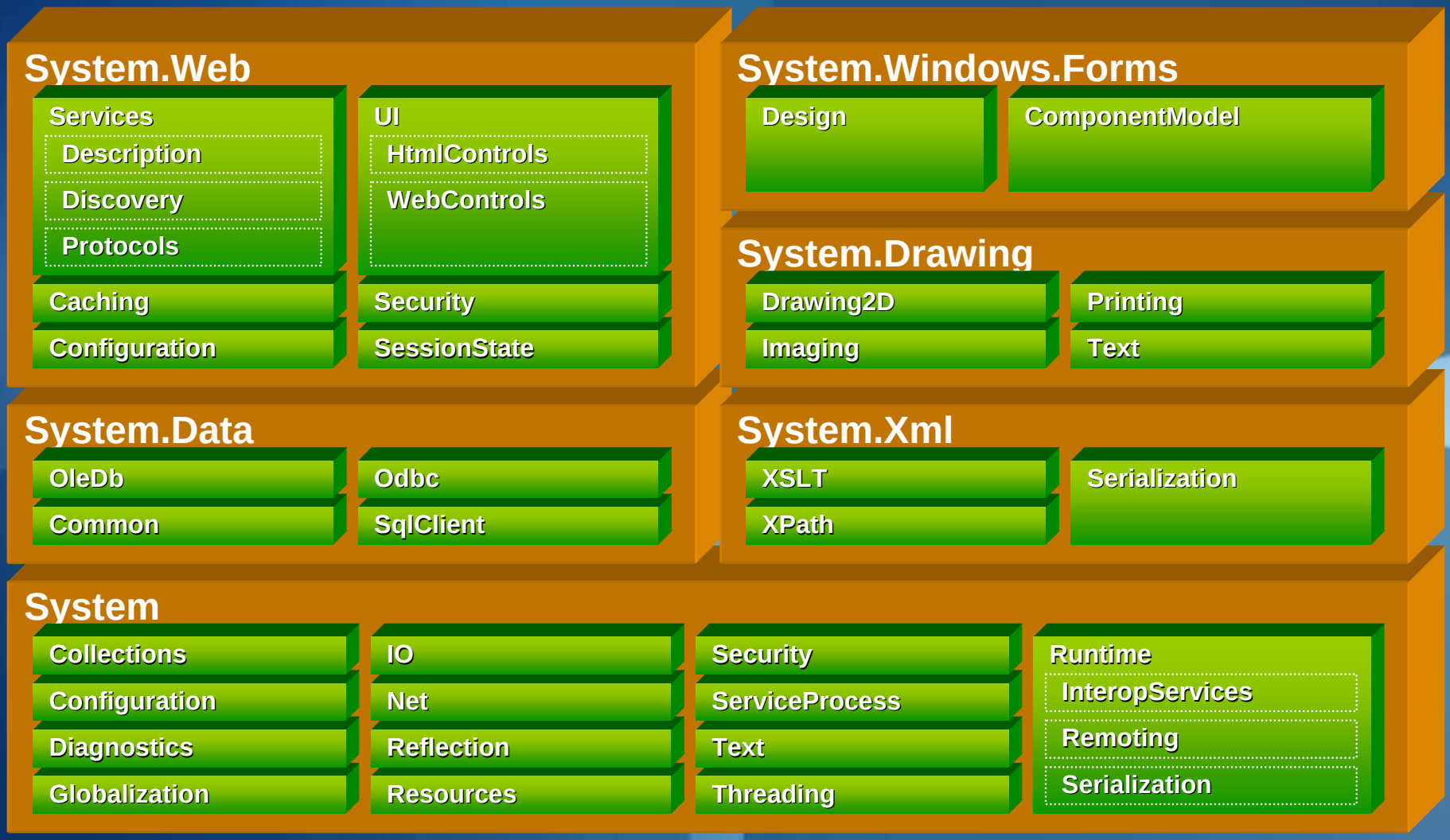
- Uno o más Assemblies
- Al ejecutar una aplicación, ¿cómo ubico los assemblies necesarios?
  - El Class Loader busca en el directorio local (preferido)
  - Global Assembly Cache (GAC)
- Diferentes aplicaciones pueden usar diferentes versiones
  - Actualizaciones más simples
  - Desinstalación más simple

.NET Framework Class Library

- Conjunto de Tipos básicos (clases, interfaces, etc.) que vienen incluídos en el .NET Framework
- Los tipos están organizados en jerarquías lógicas de nombres, denominados NAMESPACES
- Los tipos son INDEPENDIENTES del lenguaje de desarrollo
- Es extensible y totalmente orientada a objetos

.NET Framework Class Library

- El namespace raíz es **SYSTEM**



Common Language Specification (CLS)

- Especificación que estandariza una serie de características soportadas por el CLR
- Contrato entre diseñadores de lenguajes de programación y autores de bibliotecas
- Permite la interoperabilidad entre lenguajes
- Microsoft provee implementaciones de 4 lenguajes, todos compatibles con CLS
  - Microsoft Visual Basic .NET
  - Microsoft Visual C# .NET
  - Microsoft Visual J#.NET
  - Microsoft Visual C++.NET

Common Language Specification (CLS)

- El resto de la industria y el sector académico han desarrollado más de 20 lenguajes compatibles con la especificación CLS

C++.NET Visual Basic.NET C# J#

Delphi Java PHP Perl Python JavaScript

Pascal Haskell LISP Prolog RPG

Oberon Mondrian Smalltalk Eiffel ML Scheme

Cobol Fortran APL Objective Caml Mercury

CLS - Elección del lenguaje

- .NET posee un único runtime (el CLR) y un único conjunto de bibliotecas para todos los lenguajes
- No hay diferencias notorias de performance entre los lenguajes provistos por Microsoft
- El lenguaje a utilizar, en gral., dependerá de su experiencia previa con otros lenguajes o de gustos personales
  - Si conoce Java, Delphi, C++, etc. → C#
  - Si conoce Visual Basic o VBScript → VB.NET
- Los tipos de aplicaciones .NET son INDEPENDIENTES del lenguaje que elija

Temas a Tratar

- Introducción a Microsoft .NET
- Componentes Fundamentales
- Funcionamiento Interno del CLR
  - Especificación CLI
  - Modelo de Ejecución
  - Application Domains
  - Common Type System

Infraestructura de Lenguaje Común (CLI)

- Especificación patrocinada por Microsoft, Intel, HP y estandarizada por ECMA (2001) e ISO (2003) que describe:
  - Entorno de Ejecución de Aplicaciones
  - Conjunto de Librerías Básicas (BCL)
  - Tipos de Datos Comunes (CTS)
- El .NET Framework y el .NET Compact Framework son implementaciones de la especificación CLI

Sub-Especificaciones de CLI

Lenguajes de Alto Nivel

se ajustan a las reglas de la...

CLS (Common Language Specification)

y utilizan las clases de la...

BCL (Base Class Library)

cuyos tipos básicos forman el...

CTS (Common Type System)

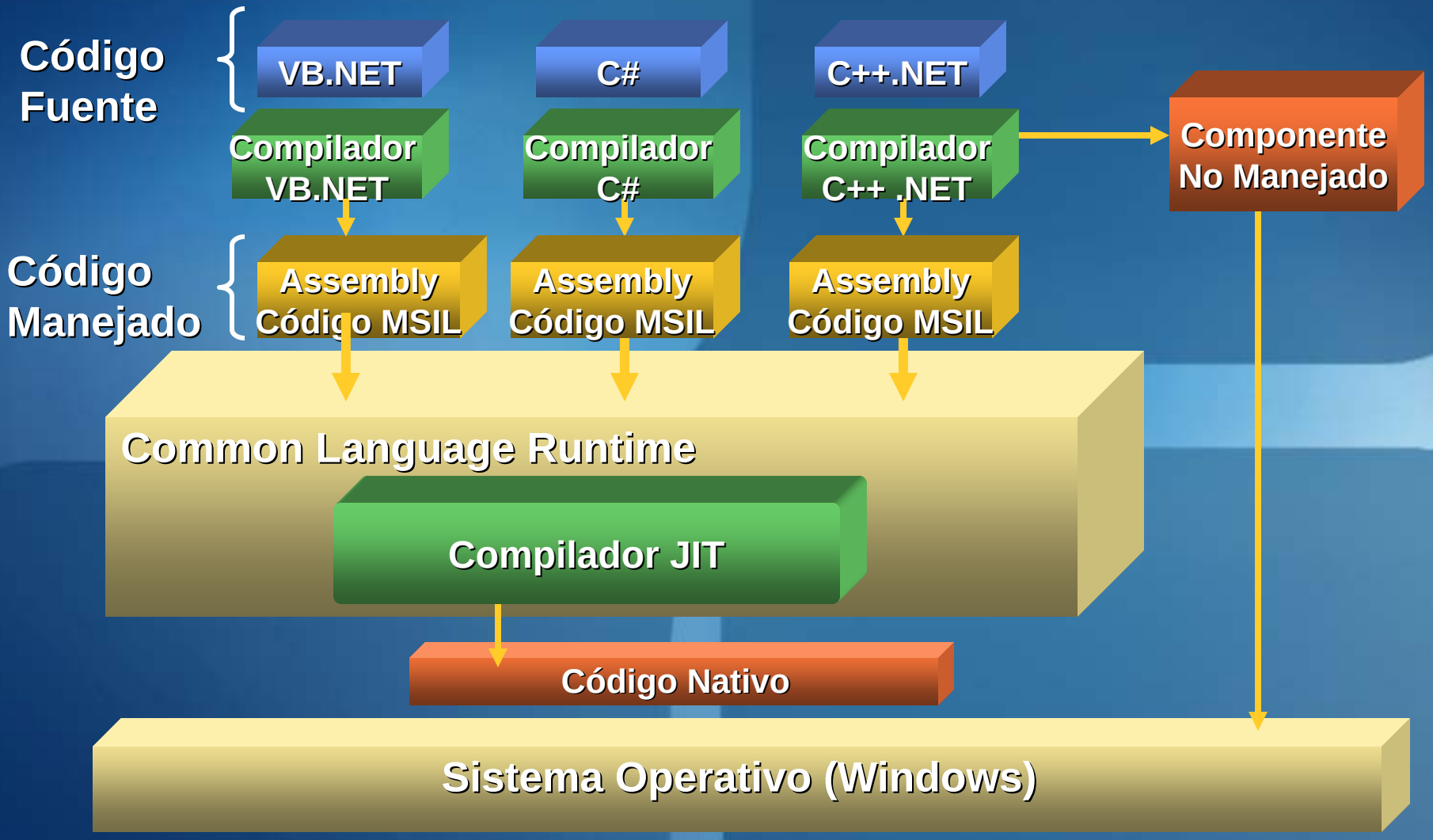
y se ejecutan bajo el control de y usan los servicios del...

CLR (Common Language Runtime)

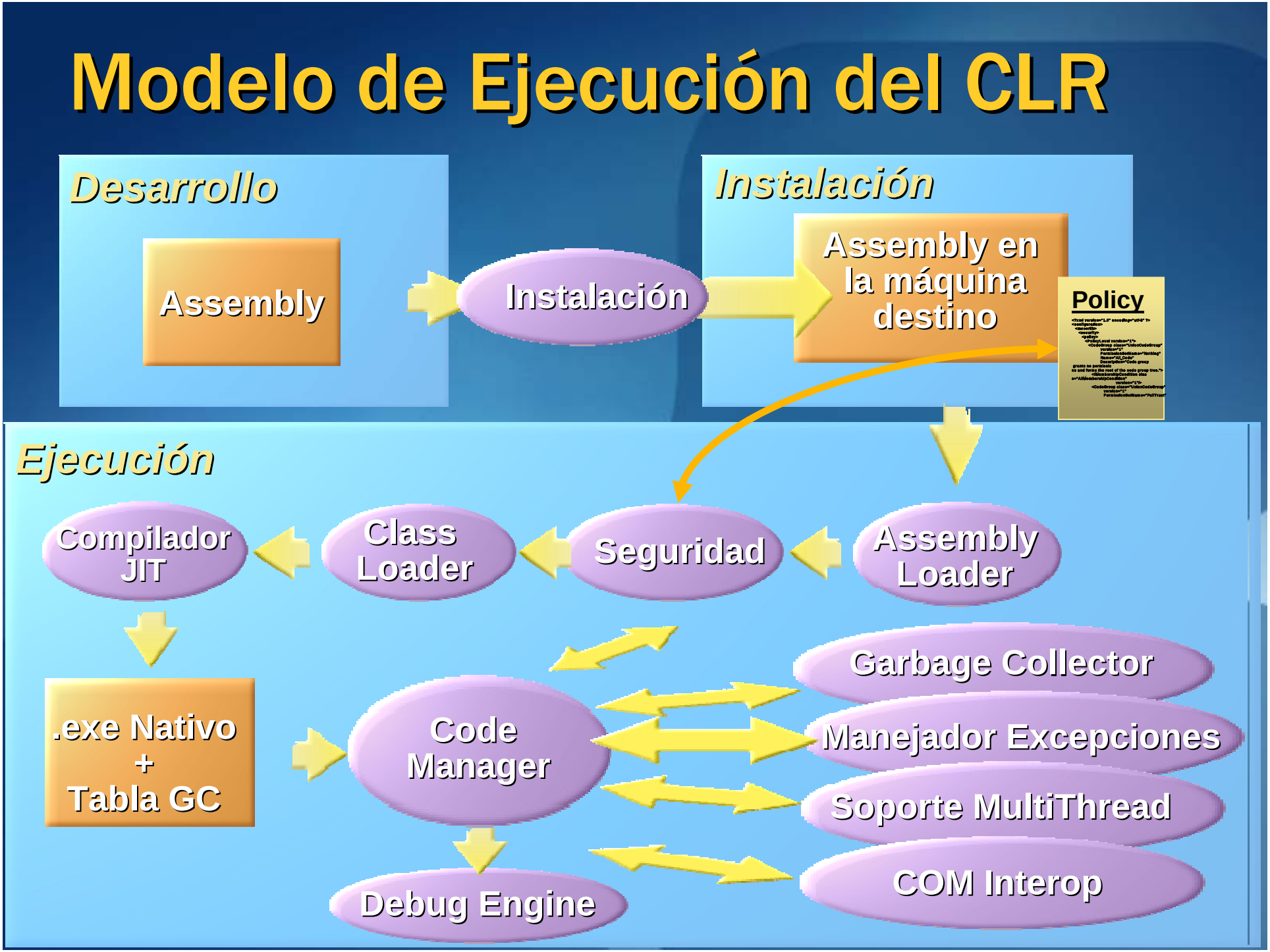
que está acoplado al y utiliza los servicios del ...

Sistema Operativo

Modelo de Ejecución del CLR



Modelo de Ejecución del CLR



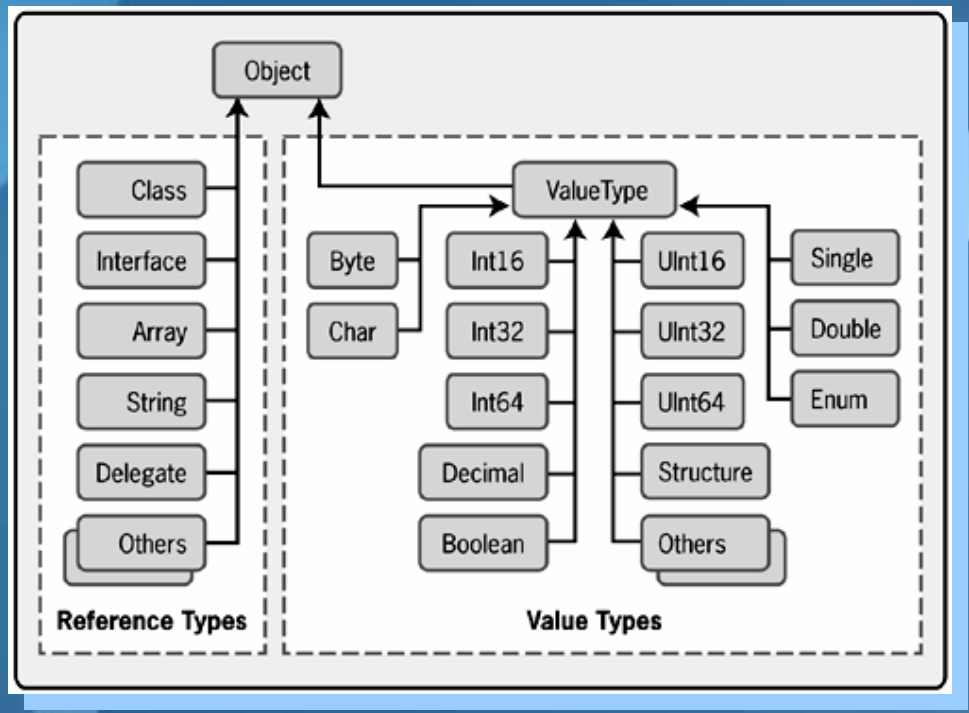
Application Domains

- Procesos virtuales dentro del CLR
  - Se ejecutan dentro de un proceso del Sistema Operativo
  - Un proceso del sistema operativo puede contener varios AppDomains
  - Más eficiente que múltiples procesos del sistema operativo
  - Más eficiente en el intercambio de contexto de ejecución
- Un Assembly y sus tipos son siempre cargados dentro de un AppDomain
- Provee una frontera para: Fallos, Tipos, Seguridad

Application Domains - CLR Host

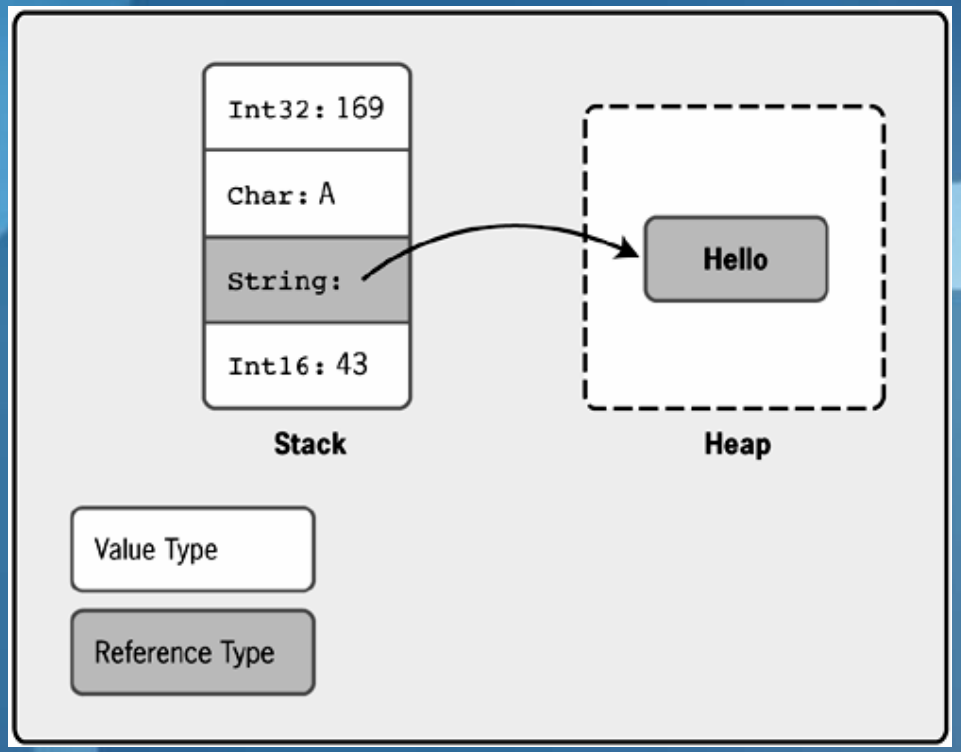
CTS (Common Type System)

- Define un conjunto común de “tipos” de datos orientados a objetos
- Todo lenguaje de programación .NET debe implementar los tipos definidos por el CTS
- Todo tipo hereda directa o indirectamente del tipo System.Object
- Define Tipos de VALOR y de REFERENCIA



La Memoria y los Tipos de Datos

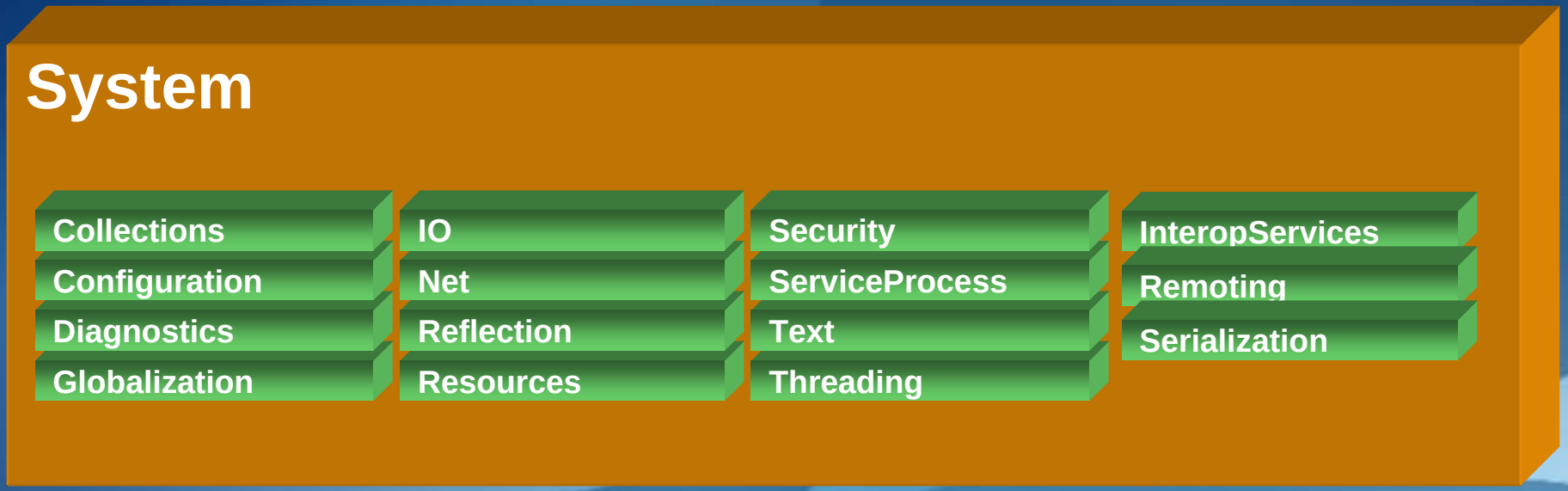
- El CLR administra dos segmentos de memoria: Stack (Pila) y Heap (Montón)
- El Stack es liberado automáticamente y el Heap es administrado por el GC (Garbage Collector)
- Los tipos VALOR se almacenan en el Stack
- Los tipos REFERENCIA se almacenan en el Heap



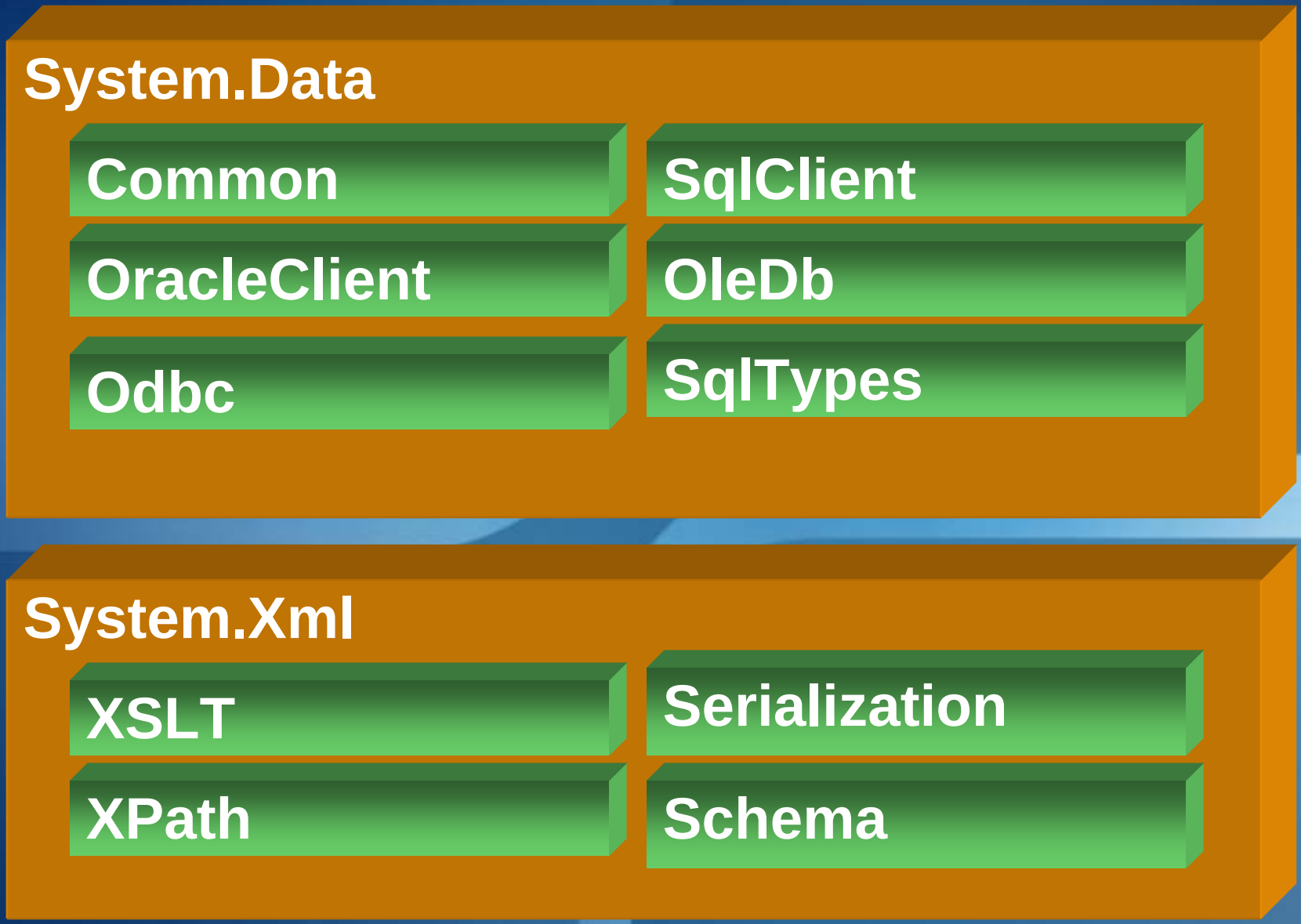
Temas a Tratar

- Introducción a Microsoft .NET
- Componentes Fundamentales
- Funcionamiento Interno del CLR
- Bibliotecas Principales
  - Base Class Library (BCL)
  - ADO.NET
  - Windows Forms
  - ASP.NET

Base Class Library



Acceso a Datos: ADO.NET



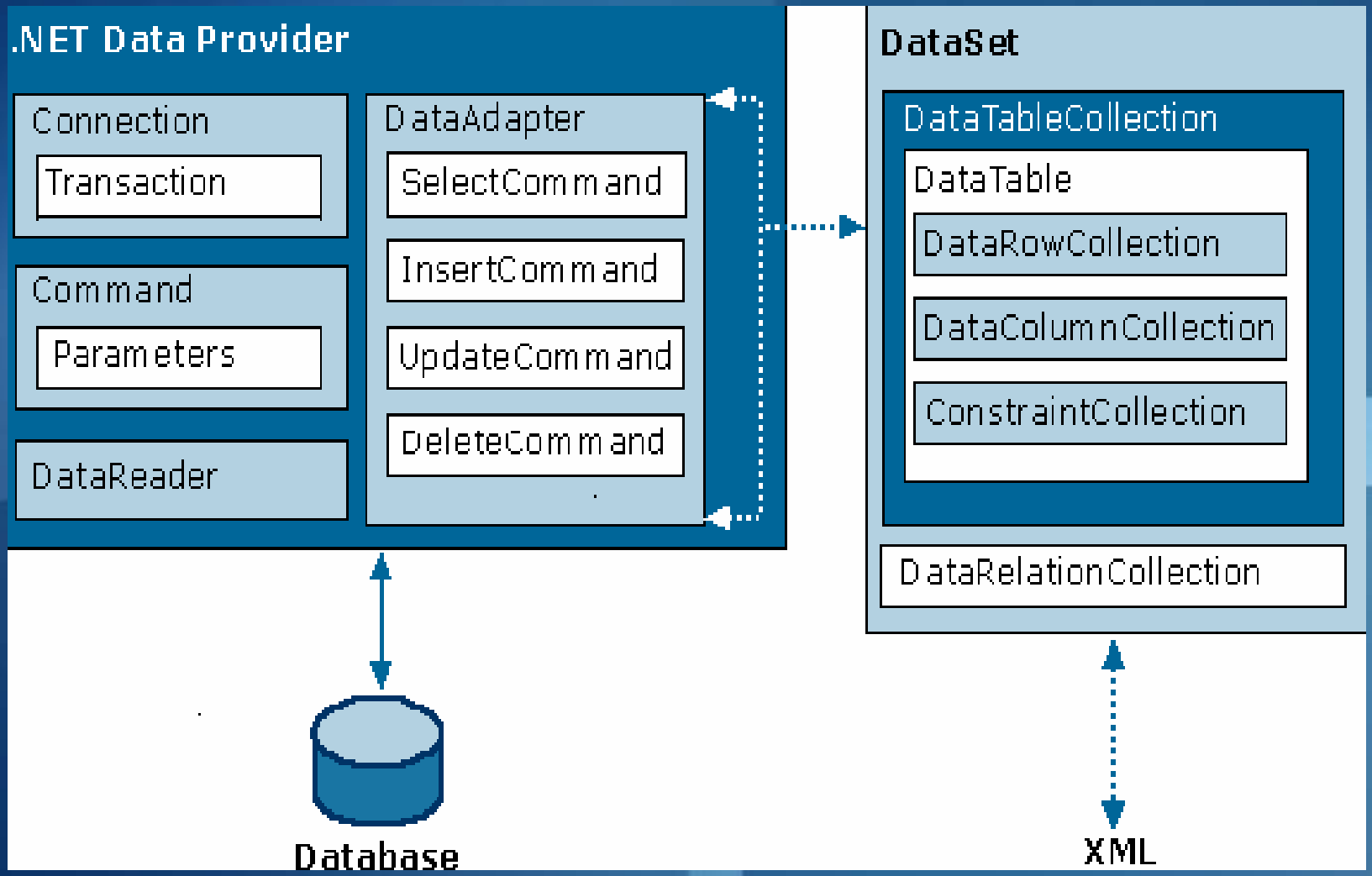
Acceso a Bases de Datos Relacionales Escenario Conectado

- Un entorno conectado es uno en el cual los usuarios están constantemente conectados a la fuente de datos
- Ventajas:
  - Mayor seguridad
  - Mejor control de concurrencia
  - Los datos se mantienen actualizados
- Desventajas:
  - Se requiere una conexión constante (consume recursos del servidor)
  - Escalabilidad

Acceso a Bases de Datos Relacionales Escenario Desconectado

- En un entorno desconectado, una parte de los datos del repositorio central se copia y modifica en forma local, para luego sincronizarse con éste.
- Ventajas
  - Se puede trabajar en forma independiente
  - Mayor escalabilidad y performance
- Desventajas
  - Los datos no están sinconizados
  - Resolución manual de conflictos

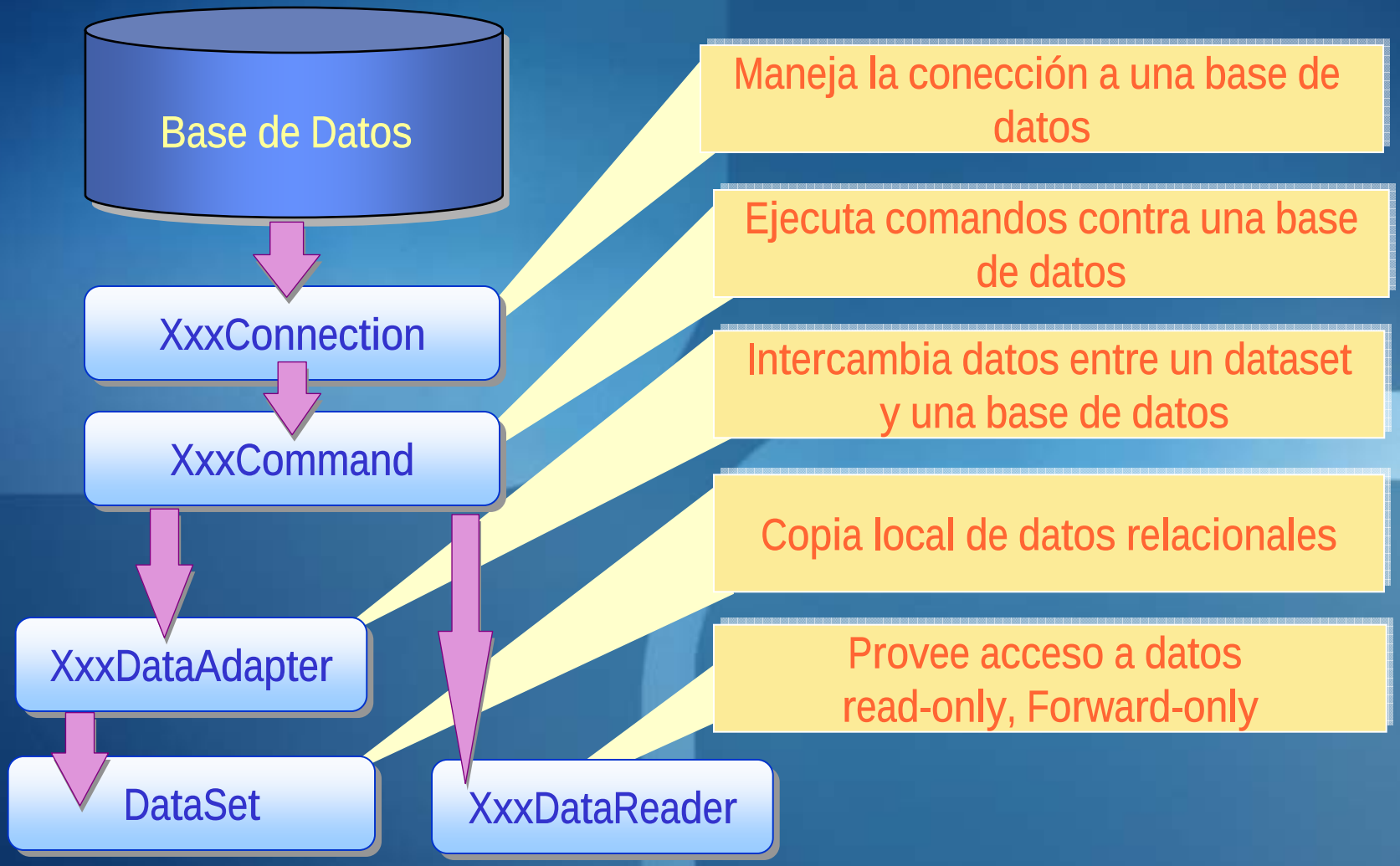
ADO.NET - Arquitectura



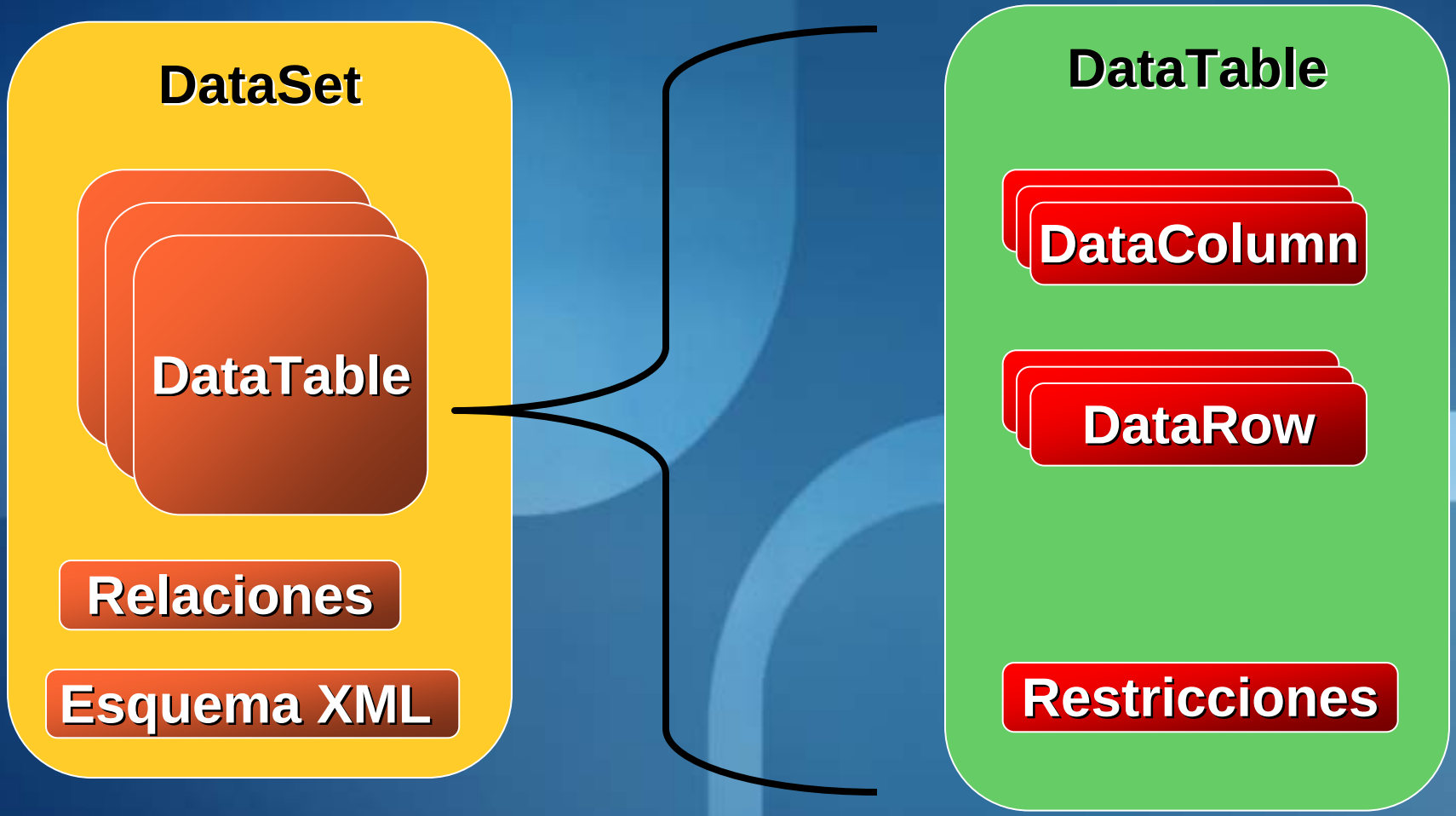
ADO.NET- Proveedores de Acceso a Datos

- SQL Server/Access (System.Data.SqlClient)
- OLE DB (System.Data.OleDb)
- ODBC (System.Data.Odbc)
- Oracle (System.Data.OracleClient)
- Otros provistos por terceros (MySQL, PostgreSQL, DB2, etc..)

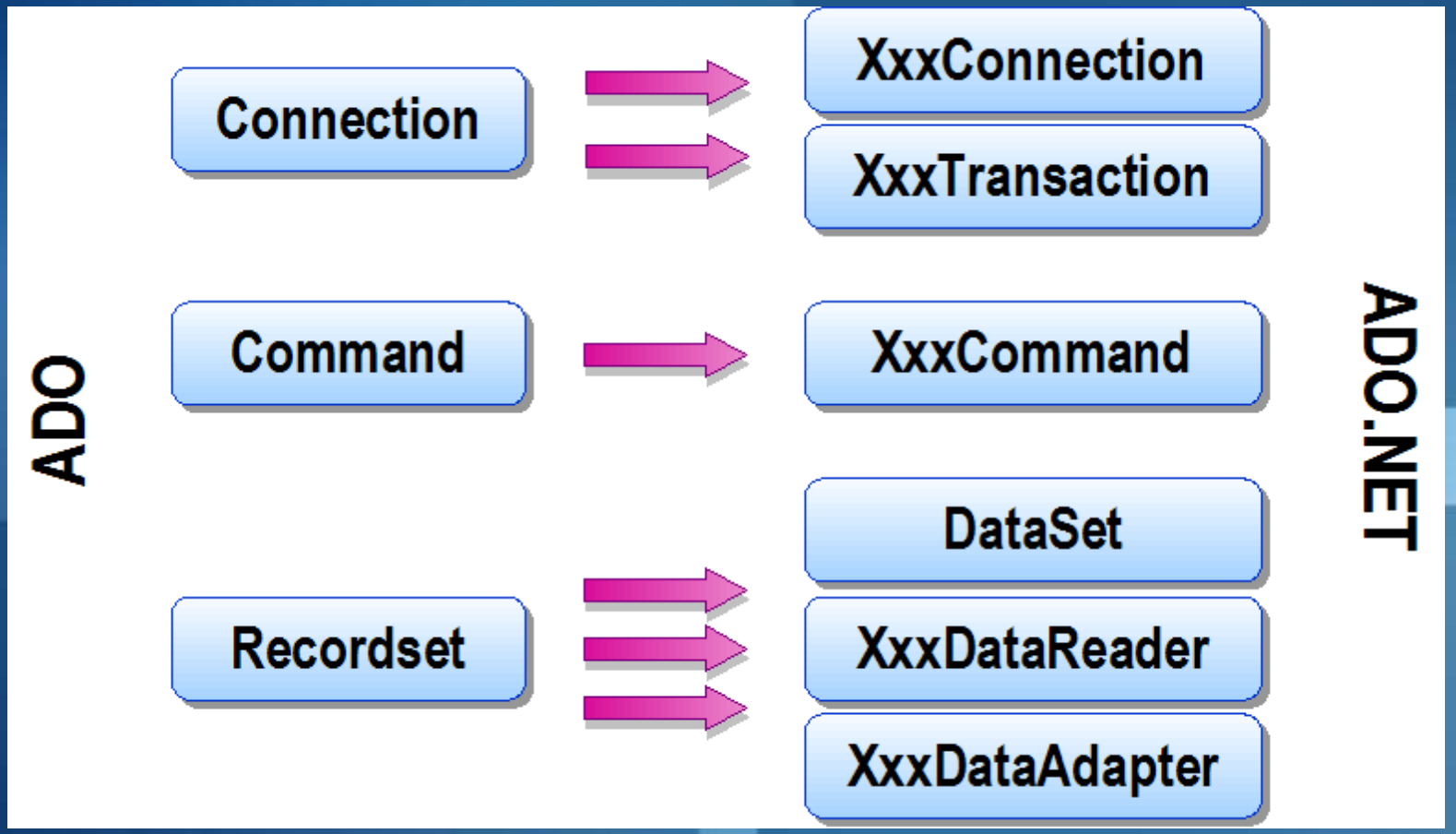
ADO.NET- Clases más comunes



ADO.NET- DataSet

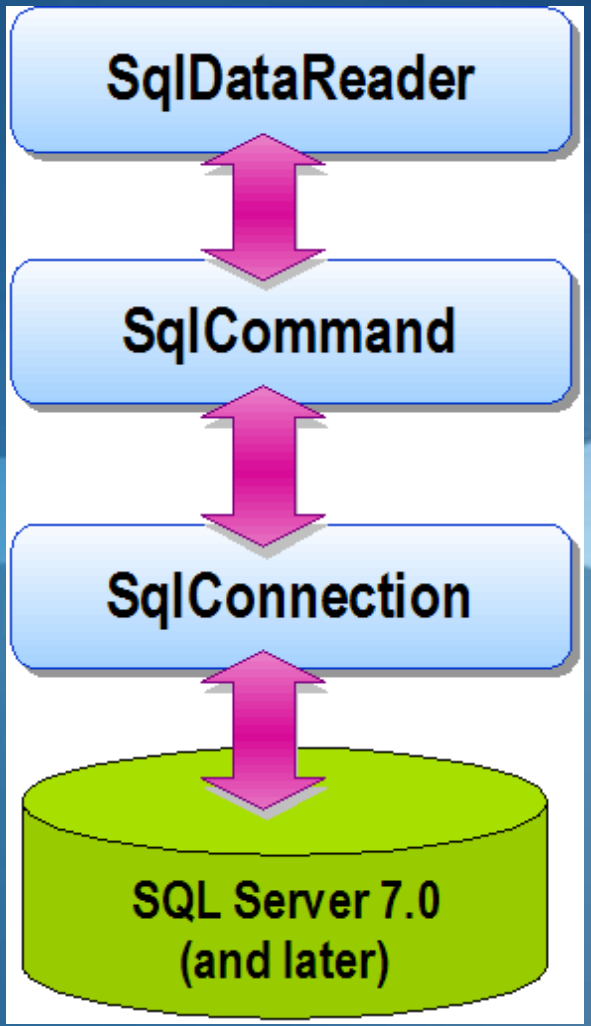


ADO.NET vs. ADO



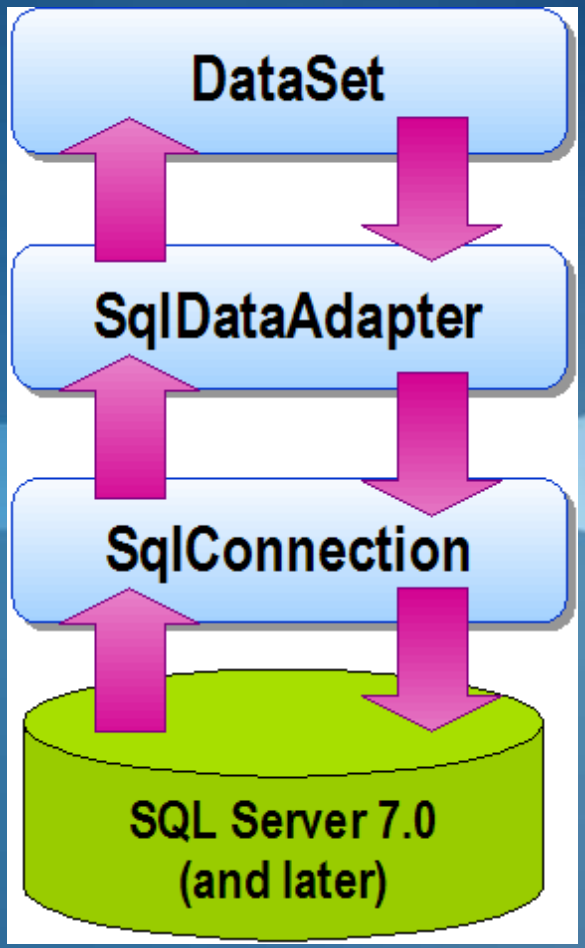
ADO.NET – Accediendo a datos Conectado

- En un escenario conectado, los recursos se mantienen en el servidor hasta que la conexión se cierra
- 1) Abrir Conexión
- 2) Ejecutar Comando
- 3) Procesar Filas en DataReader
- 4) Cerrar Reader
- 5) Cerrar Conexión

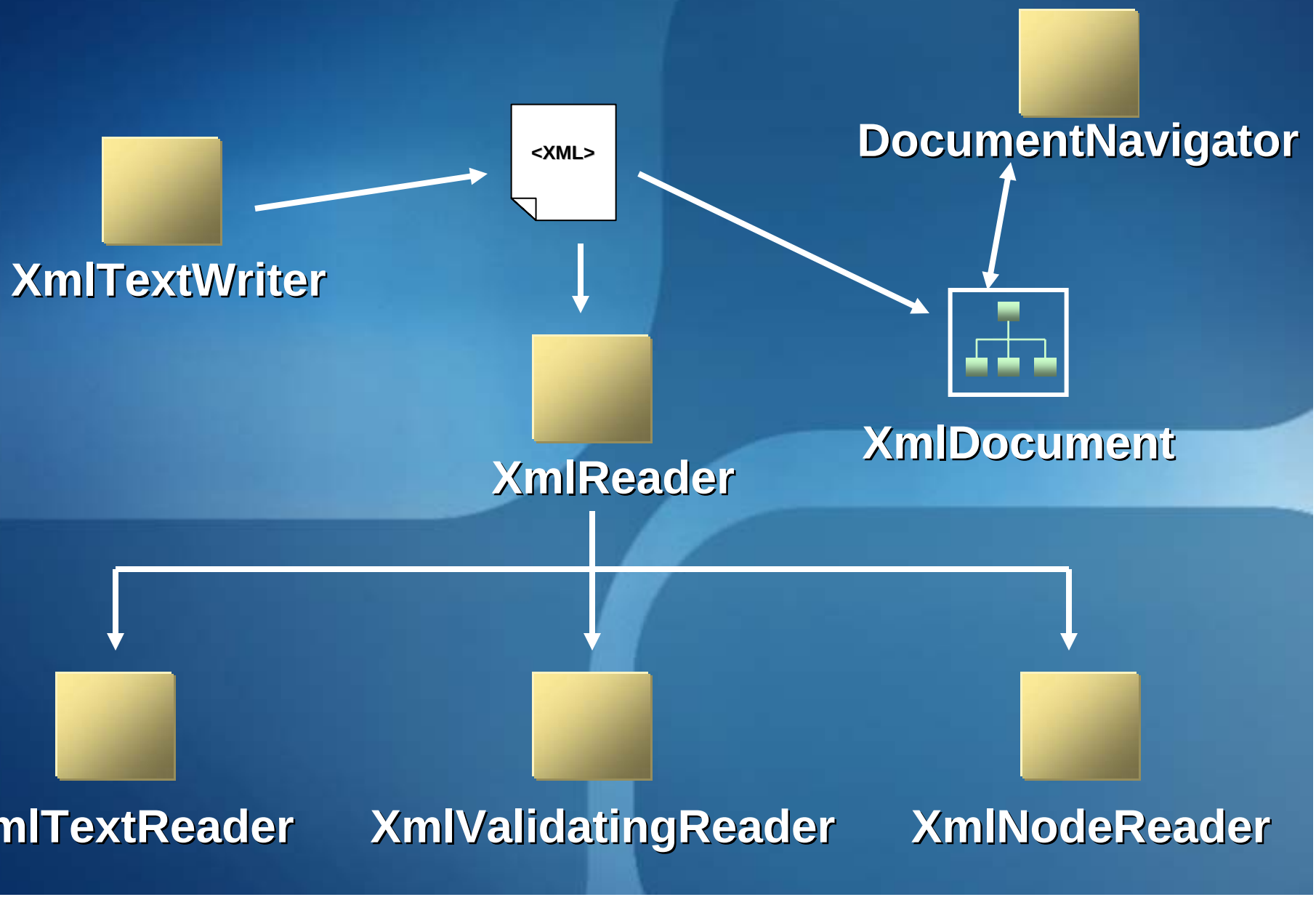


ADO.NET – Accediendo a datos Desconectado

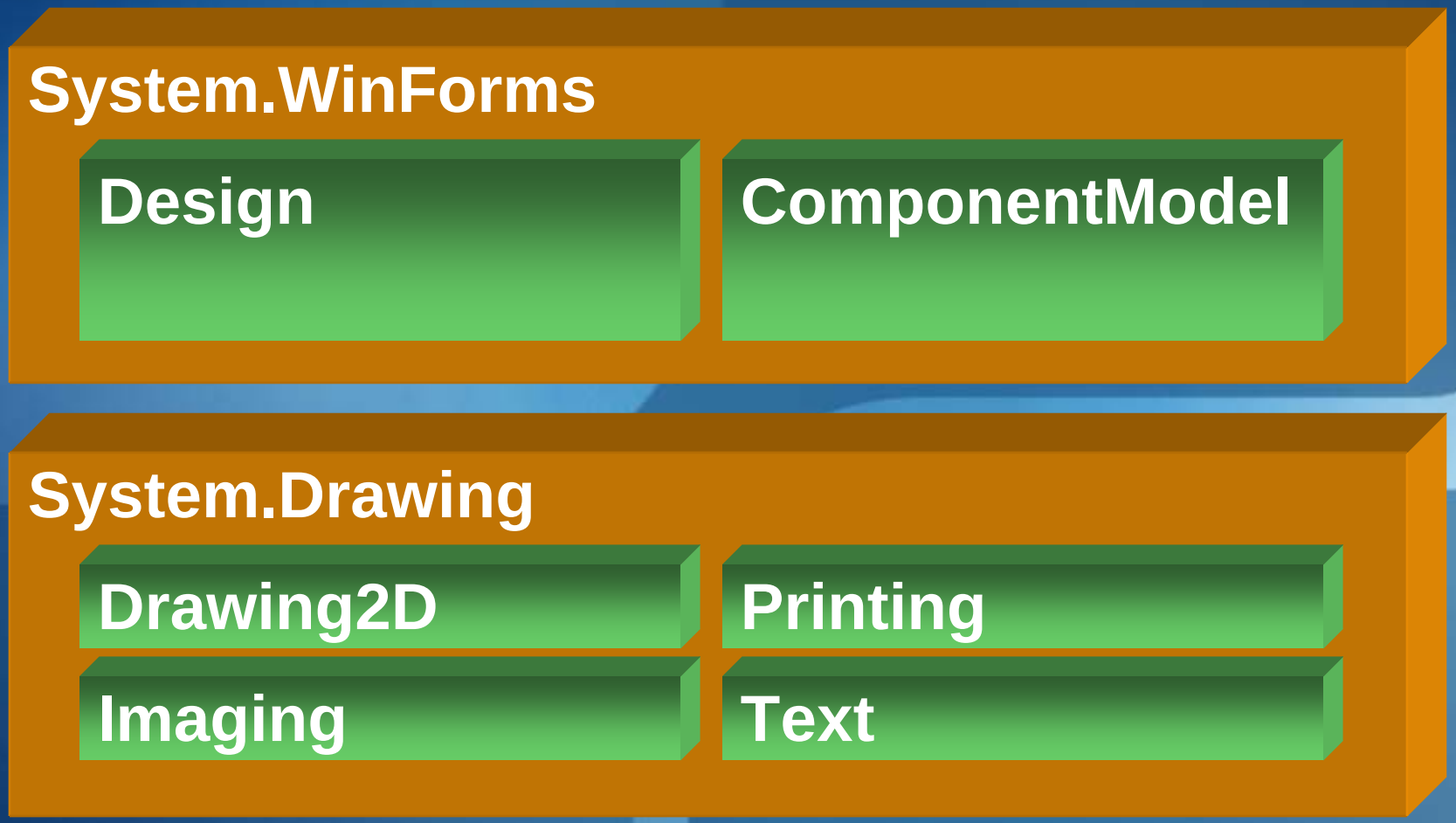
- En un escenario desconectado, los recursos no se mantienen en el servidor mientras los datos se procesan
- 1) Abrir Conexión
- 2) Llenar DataSet mediante DataAdapter
- 3) Cerrar Conexión
- 4) Procesar DataSet
- 5) Abrir Conexión
- 6) Actualizar fuente de datos mediante DataAdapter
- 7) Cerrar Conexión



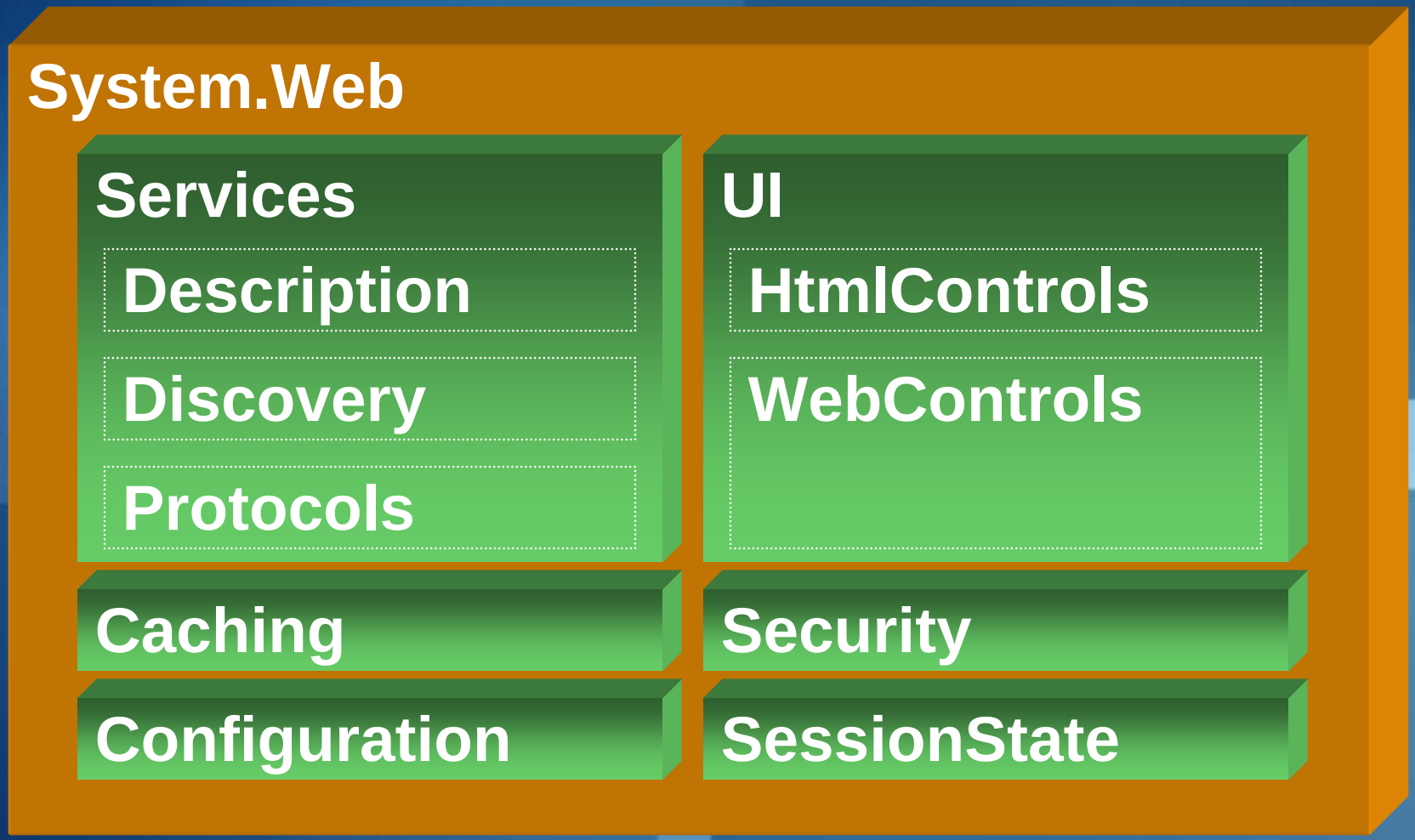
ADO.NET - Soporte a XML



Windows Forms



Aplicaciones Web: ASP.NET



Temas a Tratar

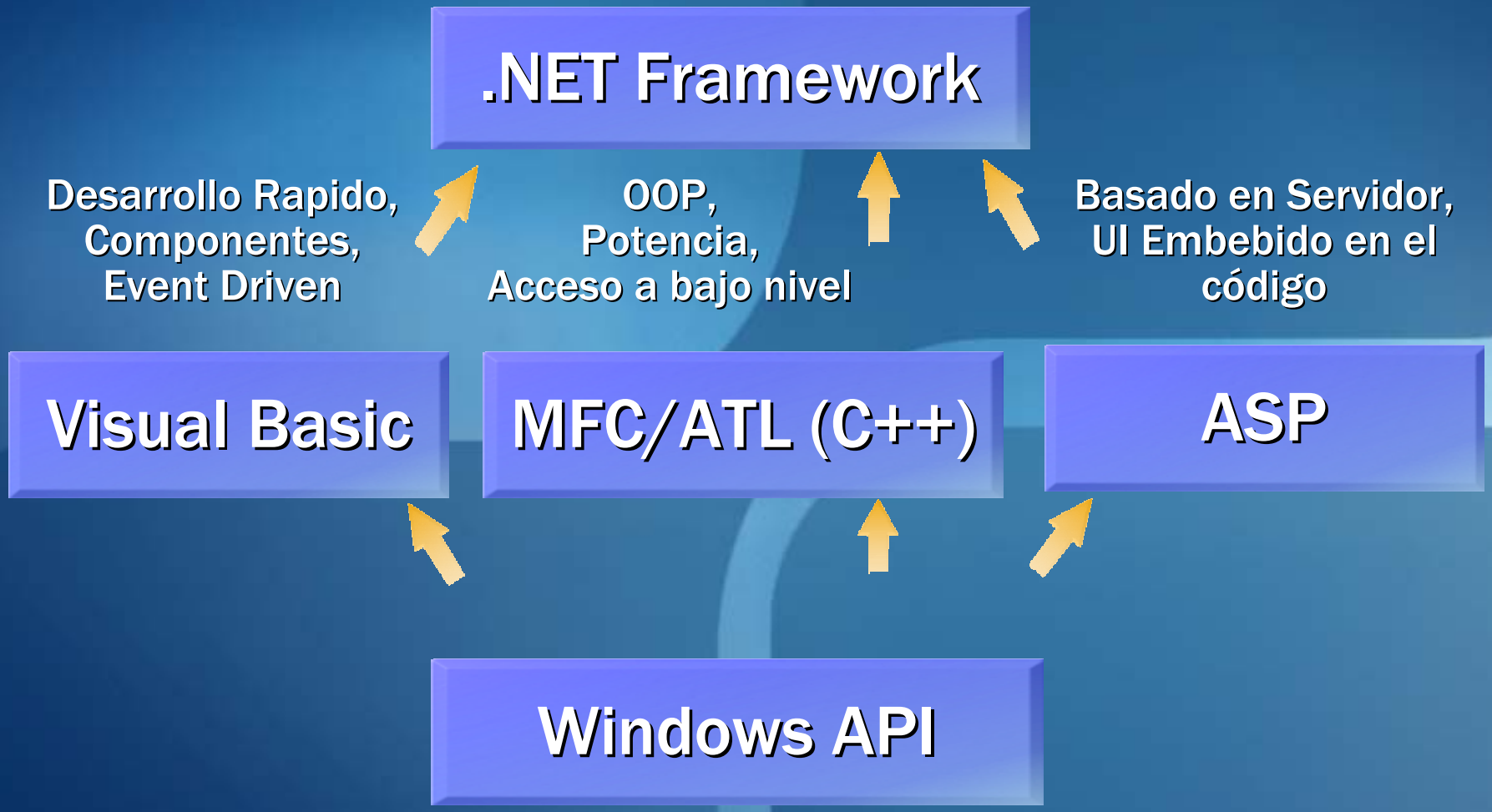
- Introducción a Microsoft .NET
- Componentes Fundamentales
- Funcionamiento Interno del CLR
- Bibliotecas Principales
- Ventajas de .NET

Ventajas de .NET

- Unifica los modelos de programación
- Simplifica aún más el desarrollo
- Provee un Entorno de Ejecución robusto y seguro
- Es independiente del lenguaje de programación
- Interoperabilidad con código existente
- Simplifica la instalación y administración de las aplicaciones
- Es Extensible

Unificando los Modelos

API consistente mas allá del lenguaje
o del modelo de programación



Desarrollo Simplificado

- Alto nivel de abstracción
  - No mas accesos COM a bajo nivel
  - Orientado a Objetos desde el Núcleo
- Sistema de tipos unificado (CTS)
  - Todo es un objeto, no mas variants
- Componentes de Software
  - Propiedades, métodos, eventos, y atributos incluidos en la construcción de clases
- API organizada en forma Jerárquica

Entorno de Ejecución Robusto y Seguro

- Gestión automática de la memoria
  - Todos los objetos son administrados por el Garbage Collector
- Manejo de Excepciones
- Fuertemente tipado
  - Solo casteos seguros
  - Inicialización de variables obligatoria
- Instalación con Cero Impacto
  - No requiere registración en la Registry

Independencia del lenguaje

- Libertad en la elección del lenguaje
  - Todas las facilidades de la plataforma .NET están disponibles a todos los lenguajes de programación .NET
  - Los componentes de una aplicación .NET pueden ser escritos en distintos lenguajes de alto nivel compatibles con la plataforma
- Herramientas compartidas
  - Debuggers, profilers, analizadores de código, y otras trabajan para todos los lenguajes

Instalación y Administración más simples

- Unidades de Ensamblado (“Assemblies”)
  - Mínima unidad de distribución, versionado y administración de seguridad de aplicaciones .NET
  - Auto-descriptas a través de un manifiesto (“manifest”)
- Instalaciones Cero-impacto
  - Aplicaciones y componentes pueden ser compartidas o privadas
- Versioning
  - Múltiples versiones del mismo componente pueden co-existir, aún en el mismo proceso

Extensibilidad

- El Framework no es una “caja negra”
- Sus clases pueden ser extendidas a través del mecanismo de herencia
  - A diferencia de COM, usamos y extendemos las clases en si mismas, no un “wrapper”
- Herencia entre distintos lenguajes

Interoperabilidad

Servicios Web XML

.NET Framework

C#

C++

VB

Otros...

COM INTEROP

COM

.NET

Temas a Tratar

- Introducción a Microsoft .NET
- Componentes fundamentales
- Funcionamiento interno
- Bibliotecas Principales
- Ventajas de .NET
- Herramientas de Desarrollo .NET
  - Visual Studio 2005
  - SQL Server 2005 Express

Visual Studio 2005

Testers

Arquitectos

Desarrolladores

Consultores

Profesionales

VB6 Devs

Part-Timers

Hobbyists

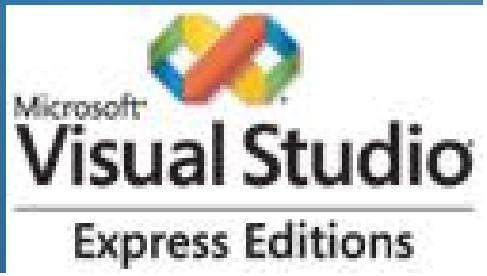
Estudiantes

Entusiastas

Novatos

Visual Studio 2005 Express Editions

- Herramientas de desarrollo gratuitas
- Muy livianas, fáciles de usar y de aprender
- Diseñadas para entusiastas, estudiantes y desarrolladores principiantes
- Hay varias ediciones, según el tipo de aplicación y el lenguaje
- Tiene características avanzadas, por ejemplo:
  - Refactoring
  - Code Snippets
  - Diseñadores WYSIWYG
  - Depuradores
  - Intellisense



SQL Server 2005 Express

- La nueva versión de MSDE
- Es gratuito
- Preparado para trabajar integrado con .NET 2.0
- Tamaño máximo de base: 4 Gb
- Max. Conexiones Concurrentes: no tiene
- Incluye una interfaz de administracion y un editor de consultas
- Mayor integración con Visual Studio 2005

Temas a Tratar

- Introducción a Microsoft .NET
- Componentes fundamentales
- Funcionamiento interno
- Bibliotecas Principales
- Ventajas de .NET
- Herramientas de Desarrollo .NET
- Novedades en .NET 2.0

Temas a Tratar

- Novedades en .NET 2.0
  - Generics
  - Soporte para 64 bits
  - Tipos Parciales
  - Nivel de Accesibilidad de Properties
  - Novedades en ADO.NET 2.0

Generics

- Son tipos parametrizados soportados por el CLR
  - Un tipo parametrizado es aquel que puede definirse sin especificar los tipos de datos de sus parámetros en tiempo de compilación.
- Nos dan la posibilidad de declarar clases, estructuras, métodos e interfaces que actuarán uniformemente sobre valores cuyos tipos se desconocen a priori y son recién especificados al momento de su utilización

Generics - Ejemplo

- Definiendo una clase genérica en C#

```
public class ClaseGenerica<T>
{
    public T atributo;
}
```

- Utilizando una clase genérica en C#

```
ClaseGenerica <string> g = new ClaseGenerica<string>();
g.atributo = “Un string";
g.atributo = 2;   //Genera Error de Compilación
…
ClaseGenerica<int> g2 = new ClaseGenerica<int>();
g2.atributo = 2; //NO genera error de compilación
```

Generics - Ejemplo

- Definiendo una clase genérica en VB.NET

```
Public Class ClaseGenerica(Of T)
    Public atributo As T
End Class
```

- Utilizando una clase genérica en VB.NET

```
Dim g As New ClaseGenerica(Of String)
g.atributo = “Un string"
g.atributo = 2   ‘Genera error de compilación
…
Dim g2 As New ClaseGenerica(Of Integer)
g2.atributo = 2 ‘NO genera error de compilación
```

Generics - Colecciones

- Colecciones Genéricas Vs. Colecciones Tradicionales

System.Collections.Generic	System.Collections
Comparer<T>	Comparer
Dictionary<K,T>	HashTable
List<T>	ArrayList
Queue<T>	Queue
SortedDictionary<K,T>	SortedList
Stack<T>	Stack
ICollection<T>	ICollection
IComparable<T>	System.IComparable
IComparer<T>	IComparer
IDictionary<K,T>	IDictionary
IEnumerable<T>	IEnumerable
IEnumerator<T>	IEnumerator
IKeyComparer<T>	IKeyComparer
IList<T>	IList

Generics - Colecciones

Sin generics

```
System.Collections.ArrayList listaTexto = new System.Collections.ArrayList();
listaTexto.Add("Un String");//Como recibe un System.Object, puedo insertar cualquier cosa
listaTexto.Add(2);  //No arroja error de compilación, pero es un error logico

for (int i = 0; i < listaTexto.Count; i++)
{
    System.Console.Write((string)listaTexto[i]); //Tengo que castear los elementos al obtenerlos de la lista
    // Se Produce un error de cast inválido al tratar de convertir el numero a un string
}
```

Con generics

```
//Utilizo la clase generica List <T> para crear una colección fuertemente tipada de strings
System.Collections.Generic.List<string> listaTexto = new System.Collections.Generic.List<string>();
listaTexto.Add("Un String"); //No hace casteo a System.Object, ya que el método Add espera un string
listaTexto.Add(2); //Arroja error de compilación, ya que sólo se pueden insertar datos de tipo string
for (int i = 0; i < listaTexto.Count; i++)
{
    System.Console.Write(listaTexto[i]); //NO Tengo que castear los elementos al obtenerlos de la lista
    //No se produce ningun error
}
```

Generics - Colecciones

VB.NET

Sin generics

```
Dim listaNumeros As New System.Collections.ArrayList()
listaNumeros.Add(2) 'Como recibe un System.Object, puedo insertar cualquier cosa
listaNumeros.Add("Un String") 'No arroja error de compilación, pero es un error logico

For i As Integer = 0 To listaNumeros.Count - 1
    'Tengo que castear los elementos al obtenerlos de la lista
    Dim tmp As Integer = System.Convert.ToInt32(listaNumeros(i))
    'Se Produce un error de cast inválido al tratar de convertir el string a un número entero
Next
```

Con generics

```
'Utilizo la clase generica List (Of T) para crear una colección fuertemente tipada de enteros
Dim listaNumeros As New System.Collections.Generic.List(Of Integer)
listaNumeros.Add(2) 'No hace casteo a System.Object, ya que el método Add espera un Integer
listaNumeros.Add("Un String") 'Arroja error de compilación, ya que sólo se pueden insertar datos de tipo entero

For i As Integer = 0 To listaNumeros.Count - 1
    'No Tengo que castear los elementos al obtenerlos de la lista
    Dim tmp As Integer = System.Convert.ToInt32(listaNumeros(i))
    'No se produce ningun error
Next
```

Soporte para 64 bits

- El CLR 1.x sólo tiene soporte para aplicaciones de 32 bits
  - No aprovechan las características de los sistemas operativos de 64 bits, ya que se ejecutan emuladas
- El CLR 2.0 tiene una versión de 64 bits
  - Permite compilar aplicaciones para que hagan uso nativamente de las nuevas características de los sistemas operativos y procesadores de 64 bits

Tipos Parciales

- Permiten la declaración de un tipo en varios archivos físicos
  - Válido para clases y estructuras
  - Válido para interfaces sólo en C#
  - Utilizan la palabra clave “partial” en la declaración
- Su uso puede tener varias ventajas
  - Dividir implementaciones complejas en partes pequeñas
  - Separación de código auto-generado
  - Mútiples desarrolladores pueden trabajar sobre distintas secciones del mismo tipo simultáneamente
  - Puede facilitar el mantenimiento y el control de versiones de código

Tipos Parciales - Ejemplo


```
// Demo.Part1.cs
using System;
public partial class Demo
{
   public Demo()
   {
      Console.Write( "P1" );
   }
}
// Demo.Part2.cs
public partial class Demo
{
   private int i;
}
```

```
// Demo.Part3.cs
// Error 1!
public class Demo
{
   // Error 2!
   private int i;
   // OK
   private int j;

   public void Test()
   {
      // Error 3!
      Console.Write( "P3" );
   }
}
```

Tipos Parciales - Ejemplo


```
' Demo.Part1.vb
Imports System

Partial Public Class Demo
    Public Sub New()
        Console.Write("P1")
    End Sub
End Class

' Demo.Part2.vb
Partial Public Class Demo
    Private i As Integer
End Class
```

```
' Demo.Part3.vb
' OK en VB.NET
Public Class Demo
    ' Error 2!
    Private i As Integer
    ' OK
    Private j As Integer

    Public Sub Test()
        ' OK en VB.NET
        Console.Write("P3")
    End Sub

End Class
```

Modificadores de acceso

- El CLR 2.0 permite especificar diferentes modificadores de acceso para el get y el set de las propiedades e índices
- Permite solamente modificar uno de los dos elementos de acceso, mientras que el otro toma el nivel de acceso de la property
- Puede especificar más restricciones
- Mejora el encapsulamiento de atributos

Modificadores de acceso - Ejemplos

C#

```
public class Customer{
    private string id;
    public string CustomerId {
        public get { return id; }
        private set { id = value; }
    }
}
```

VB.NET

```
Public Class Customer
    Private id As String
    Public Property CustomerId() As String
        Get
            Return Me.id
        End Get
        Private Set(ByVal value As String)
            Me.id = value
        End Set
    End Property
End Class
```

Novedades en ADO.NET 2.0

- API independiente del proveedor ADO.NET
  - Modelada bajo el patrón “Abstract Factory”
- Operaciones Asincrónicas
  - Permite ejecutar comandos contra la base de datos de manera asincrónica no bloqueante
- Multiple Active Result Sets (MARS)
  - Permite tener múltiples DataReaders abiertos sobre la misma conexión

Novedades en ADO.NET 2.0

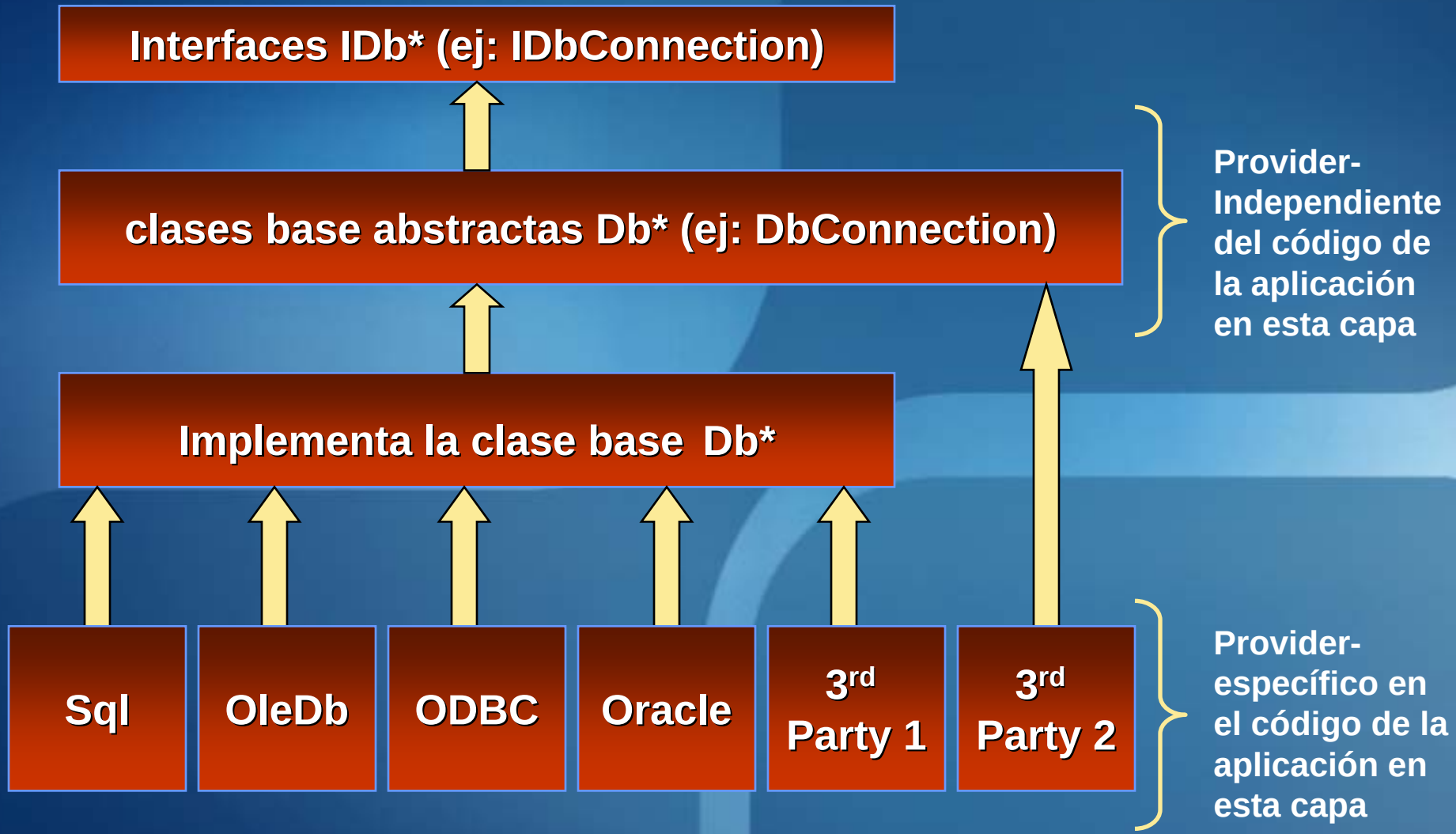
- Integración y aprovechamiento de características de SQL Server 2005
  - Notification Services
  - Service Broker
  - Tipo de dato XML
  - Soporte a Servicios Web
- Mejoras en el DataSet y DataTable
  - Performance
  - Serialización
  - Soporte a XML

ADO.NET 2.0 – API Independiente

- Namespace System.Data.Common

DbCommand	DbCommandBuilder	DbConnection
DataAdapter	DbDataAdapter	DbDataReader
DbParameter	DbParameterCollection	DbTransaction
DbProviderFactory	DbProviderFactories	DbException

ADO.NET 2.0 – API Independiente



ADO.NET 2.0 - DataSet

- Mejoras de performance
  - Mantienen indices internos de los registros de sus DataTables
- Serialización binaria del contenido
  - El DataSet 1.x es siempre serializado a XML
    - Bueno para integrar datos, malo en performance
  - El DataSet 2.0 soporta serialización binaria
    - Rápido y compacto
    - DataSet.RemotingFormat = SerializationFormat.Binary

ADO.NET 2.0 - DataTable

- Operaciones comunes del DataSet también disponibles en el DataTable:
  - ReadXml, ReadXmlSchema, WriteXml, WriteXmlSchema, Clear, Clone, Copy, Merge, GetChanges
- DataTable es auto-serializable:
  - Buen mecanismo para transmitir datos en una aplicación distribuída

ADO.NET 2.0 - Tipo de dato XML en el DataSet

- DataTable acepta columnas de tipo XML
  - System.Data.SqlTypes.SqlXml
- Expuestas como una instancia de XPathDocument
- Pueden accederse vía XmlReader
- Facilidades para trabajar con documentos XML como un conjunto de valores

ADO.NET 2.0 - Actualizaciones Batch

- ADO.NET 2.0 permite ejecutar múltiples instrucciones SQL sobre una base de datos de forma batch, usando el sp_executesql
- Reduce tráfico de red
- DataAdapter.UpdateBatchSize = batch_size
- Trabaja con transacciones
- Trabaja con los proveedores para SQL Server y Oracle

Microsoft®