**GUÍA DE REPASO CONCEPTUAL**

LENGUAJES DE ÚLTIMA GENERACIÓN

CÓDIGO DE MATERIA 22

Titular: Pablo Vilaboa

FACULTAD DE TECNOLOGÍA INFORMÁTICA

UNIVERSIDAD ABIERTA INTERAMERICANA

2020

**ÍNDICE**

GUÍA DE REPASO CONCEPTUAL

GUÍA DE EJERCICIOS

GUÍA DE ABORDAJE BIBLIOGRÁFICO

PREGUNTAS

**UNIDAD I 🡪 ENSAMBLADOS**

1. ¿Qué son los metadatos?

En el contexto de C# y ensamblados, los metadatos son información adicional que se incluye en un ensamblado para describir su contenido y facilitar la interoperabilidad y la gestión del código. Estos metadatos proporcionan detalles sobre los tipos de datos, miembros de clases, referencias a otros ensamblados y más.

Algunos ejemplos de información que se incluye en los metadatos son:

* **Información sobre tipos:** Se describen los tipos de datos, como clases, interfaces, estructuras, enumeraciones, y sus miembros (métodos, propiedades, campos, eventos).
* **Referencias a otros ensamblados:** Se incluye información sobre los ensamblados que son referenciados por el ensamblado actual. Esto permite que el compilador y el entorno de ejecución localicen las dependencias necesarias.
* **Atributos:** Se pueden agregar atributos a los tipos y miembros para proporcionar información adicional, como información de depuración, serialización, seguridad, etc.
* **Versiones y cultura:** Se pueden incluir detalles sobre la versión del ensamblado, la cultura a la que pertenece y otra información relacionada con la versión.
* **Referencias a recursos:** Si el ensamblado contiene recursos, como imágenes, texto o archivos de configuración, se incluye información sobre cómo acceder a esos recursos.

Los metadatos son cruciales para el funcionamiento adecuado de la interoperabilidad entre ensamblados, ya que permiten a la CLR (Common Language Runtime) entender y gestionar el código contenido en un ensamblado. Además, los metadatos son utilizados durante el proceso de reflexión, que permite a las aplicaciones inspeccionar y manipular tipos y miembros en tiempo de ejecución.

2. ¿Qué es el MSIL?

MSIL (Microsoft Intermediate Language), también conocido como CIL (Common Intermediate Language), es un lenguaje intermedio utilizado en el entorno de ejecución de la plataforma .NET. Cuando se compila un programa escrito en un lenguaje de alto nivel como C# o VB.NET, el compilador no genera código máquina específico para la arquitectura de la CPU de destino. En su lugar, produce MSIL, que es un código independiente de la arquitectura.

3. ¿Para qué usa el MSIL el Framework .NET?

* **Independencia de plataforma:** El código MSIL se puede ejecutar en cualquier sistema que tenga un entorno de ejecución .NET compatible, como la CLR (Common Language Runtime).
* **Just-In-Time Compilation (JIT):** Durante la ejecución de una aplicación .NET, el código MSIL se compila a código máquina nativo específico de la arquitectura de la CPU en el momento justo antes de su ejecución. Este proceso se conoce como compilación Just-In-Time.
* **Seguridad:** El código MSIL es verificado por la CLR para garantizar que cumple con las reglas de seguridad de .NET antes de ser ejecutado. Esto ayuda a prevenir vulnerabilidades y errores comunes de programación.
* **Interoperabilidad:** Diferentes lenguajes de programación que se compilan para MSIL pueden trabajar juntos de manera transparente, ya que comparten un formato común y se ejecutan en el mismo entorno de ejecución.
* **Reflexión:** La información contenida en los metadatos del ensamblado (que incluyen MSIL) permite la reflexión, lo que significa que las aplicaciones pueden inspeccionar y manipular tipos y miembros en tiempo de ejecución.

En resumen, el MSIL es una representación intermedia del código fuente de alto nivel en el entorno de ejecución .NET, permitiendo la portabilidad, seguridad y interoperabilidad entre diferentes lenguajes de programación que se compilan para la plataforma .NET.

4. ¿Qué es un ensamblado?

Un ensamblado en el contexto de la programación en el entorno .NET, que incluye lenguajes como C#, VB.NET, y otros, es un paquete que contiene código ejecutable y los metadatos necesarios para describir y administrar ese código. En esencia, un ensamblado es la unidad fundamental de implementación y reutilización en la plataforma .NET.

Un ensamblado puede contener uno o más archivos, incluyendo:

* **Código MSIL:** El código MSIL (Microsoft Intermediate Language) es el código intermedio generado por el compilador para ser ejecutado por la CLR (Common Language Runtime). Este código es independiente de la arquitectura y se compila a código nativo durante la ejecución.
* **Metadatos:** Información adicional sobre el ensamblado, incluyendo detalles sobre los tipos de datos, miembros de clases, referencias a otros ensamblados, atributos, versiones, cultura, y más. Los metadatos son esenciales para la interoperabilidad y la reflexión en .NET.
* **Recursos:** Archivos adicionales como imágenes, archivos de configuración, archivos de recursos localizables, etc., que pueden ser utilizados por la aplicación.

Existen dos tipos principales de ensamblados en .NET:

* **Ensamblados privados:** Estos son ensamblados utilizados por una aplicación específica y se almacenan en el directorio de la aplicación. Pueden incluir tanto ensamblados ejecutables (.exe) como bibliotecas de clases (.dll).
* **Ensamblados compartidos (o de biblioteca):** Estos son ensamblados destinados a ser compartidos por múltiples aplicaciones. Se almacenan en el Global Assembly Cache (GAC) y pueden ser referenciados por varias aplicaciones.

El concepto de ensamblado proporciona beneficios como la encapsulación de código, la reutilización de componentes, la gestión de versiones y la facilidad de implementación. Además, la CLR utiliza información contenida en los ensamblados, como los metadatos, para realizar tareas como la verificación de seguridad, la compilación Just-In-Time (JIT) y la reflexión en tiempo de ejecución.

5. ¿Qué es el manifiesto?

El manifiesto es una parte fundamental de un ensamblado en el entorno .NET. Se trata de un conjunto de metadatos que proporciona información esencial sobre el ensamblado y su contenido. El manifiesto se encuentra dentro del ensamblado y contiene detalles importantes que son utilizados por la Common Language Runtime (CLR) durante la carga y ejecución del programa. Algunos de los elementos clave que se encuentran en el manifiesto son:

* **Identificación del ensamblado:** El manifiesto contiene información sobre el nombre del ensamblado, su versión, la cultura a la que pertenece y la clave pública si está firmado.
* **Lista de archivos contenidos:** Enumera los archivos que forman parte del ensamblado, incluyendo el archivo principal (ya sea un ejecutable .exe o una biblioteca de clases .dll) y otros archivos como recursos.
* **Referencias a otros ensamblados:** Indica los ensamblados que son referenciados por el ensamblado actual. Esto es esencial para que la CLR pueda localizar y cargar las dependencias necesarias durante la ejecución.
* **Información sobre los permisos y la seguridad:** Incluye detalles sobre los permisos requeridos por el ensamblado y cualquier información relacionada con la seguridad.
* **Información sobre la relación con el GAC:** Si el ensamblado está destinado a ser compartido y almacenado en el Global Assembly Cache (GAC), el manifiesto contiene información sobre su relación con el GAC.

Es importante destacar que el manifiesto es parte integral de la estructura de un ensamblado en .NET y contribuye a la interoperabilidad, la gestión de versiones y la seguridad en el entorno de ejecución. Además, el manifiesto junto con otros metadatos y el código MSIL (Microsoft Intermediate Language) constituyen los componentes esenciales de un ensamblado que la CLR utiliza durante la carga y ejecución de la aplicación.

DIFERENCIA ENTRE METADATOS Y MANIFIESTO

Metadatos: Información adicional que describe el contenido del ensamblado, incluyendo detalles sobre tipos, miembros, referencias a otros ensamblados, etc.

Manifiesto: Una parte específica de los metadatos que proporciona información clave sobre la identidad y la estructura del ensamblado, como su nombre, versión, cultura, archivos contenidos, referencias a otros ensamblados, etc.

6. ¿Para qué se usan los nombres seguros?

En el contexto de .NET y los ensamblados, los "nombres seguros" se refieren al uso de nombres fuertemente tipados y únicos para identificar los ensamblados y sus componentes. Esto se logra mediante el uso de la combinación de nombre, versión, cultura y clave pública (si se aplica) en el manifiesto del ensamblado.

Los nombres seguros son esenciales por varias razones:

* **Evitar conflictos de nombres:** Al tener nombres únicos y específicos, se reducen las posibilidades de conflictos de nombres entre diferentes ensamblados. Esto es crucial, especialmente cuando se utilizan múltiples bibliotecas y componentes en una aplicación.
* **Gestión de versiones:** Los nombres seguros incluyen información de versión. Esto permite a la CLR gestionar y cargar diferentes versiones de un mismo ensamblado de forma simultánea, garantizando que las aplicaciones utilicen la versión correcta de la biblioteca o componente.
* **Seguridad:** Si se utiliza la firma de ensamblados mediante claves públicas y privadas, los nombres seguros ayudan a garantizar que el ensamblado no haya sido alterado, ya que la firma dependerá de los detalles específicos del nombre seguro.
* **Interoperabilidad:** Al tener nombres seguros, los ensamblados pueden ser referenciados de manera confiable desde otros ensamblados, facilitando la interoperabilidad entre diferentes partes de una aplicación o incluso entre aplicaciones.
* **Global Assembly Cache (GAC):** El GAC almacena ensamblados compartidos de forma global en el sistema, y los nombres seguros son fundamentales para identificar y gestionar estos ensamblados de manera efectiva.

En resumen, los nombres seguros proporcionan un sistema robusto para la identificación única y segura de ensamblados en el entorno .NET, facilitando la gestión, la interoperabilidad y la seguridad en el desarrollo de aplicaciones.

7. ¿Qué es y para qué se utiliza el GAC?

El GAC, o Global Assembly Cache (Caché Global de Ensamblados), es un repositorio centralizado en el sistema de archivos de Windows utilizado por la plataforma .NET para almacenar y gestionar versiones compartidas de ensamblados. Su propósito principal es proporcionar un lugar común y seguro para almacenar ensamblados que deben ser compartidos entre varias aplicaciones o por todo el sistema.

Características y funciones clave del GAC:

* **Almacenamiento Global:** El GAC almacena versiones compartidas de ensamblados de forma global en el sistema, lo que permite que múltiples aplicaciones accedan y utilicen la misma versión de un ensamblado.
* **Gestión de Versiones:** Permite la coexistencia de múltiples versiones de un mismo ensamblado. Cada versión del ensamblado se identifica mediante su nombre fuerte, que incluye información como el nombre del ensamblado, la versión, la cultura y, si es aplicable, el token de clave pública.
* **Evitar Conflictos de Nombres:** Al utilizar nombres fuertes, el GAC ayuda a evitar conflictos de nombres entre diferentes versiones o aplicaciones que podrían utilizar ensamblados con el mismo nombre.
* **Firma de Ensamblados:** Los ensamblados almacenados en el GAC a menudo están firmados digitalmente mediante una clave privada, proporcionando seguridad y garantizando la integridad del ensamblado.
* **Instalación y Gestión:** La instalación y gestión de ensamblados en el GAC se realiza mediante herramientas como el programa de línea de comandos Gacutil o mediante interfaces gráficas proporcionadas por herramientas de desarrollo como Visual Studio.
* **Acceso desde múltiples ubicaciones:** Los ensamblados almacenados en el GAC pueden ser accedidos por aplicaciones de diferentes ubicaciones en el sistema de archivos, lo que facilita la reutilización y la interoperabilidad.

En resumen, el GAC es un componente clave en el ecosistema .NET que proporciona un mecanismo para compartir y gestionar ensamblados de manera global, facilitando la coexistencia de múltiples versiones y la prevención de conflictos de nombres en aplicaciones que utilizan la plataforma .NET.

8. ¿Cómo puede ver el registro de enlace de ensamblados?

El registro de enlace de ensamblados en .NET es un registro que almacena información sobre cómo se resuelven las referencias a ensamblados durante la ejecución de una aplicación. Este registro puede ser útil para diagnosticar problemas relacionados con la resolución de ensamblados y para entender cómo el sistema encuentra y carga los ensamblados necesarios.

Puedes ver el registro de enlace de ensamblados de las siguientes maneras:

**Visual Studio Output Window**

* Abre Visual Studio: Si estás utilizando Visual Studio como tu entorno de desarrollo, abre tu proyecto.
* Configura la salida (Output): Ve a View -> Output para abrir la ventana de salida.
* Selecciona "Enlace de ensamblados": En el menú desplegable de la ventana de salida, selecciona "Enlace de ensamblados" o "Assembly Binding" en el filtro. Aquí verás mensajes detallados sobre cómo se resuelven las referencias de ensamblados.

**Uso de Fuslogvw (Assembly Binding Log Viewer)**

* Abre Fuslogvw: Fuslogvw.exe es una herramienta que puede ayudarte a ver el registro de enlace de ensamblados. Puedes abrirlo desde la línea de comandos ejecutando fuslogvw o buscándolo en el menú Inicio.
* Habilita el registro: En Fuslogvw, selecciona la pestaña "Log Settings" y activa la opción "Log bind failures to disk". Esto asegurará que se registren las fallas de enlace en un archivo.
* Reproduce el problema: Ejecuta tu aplicación para reproducir el problema relacionado con la resolución de ensamblados.
* Ver resultados: Vuelve a Fuslogvw y selecciona la pestaña "Log Entries". Aquí verás una lista de eventos de enlace de ensamblados, incluyendo información sobre éxitos y fallos de resolución.

**Revisando Archivos de Registro**

* El registro de enlace de ensamblados se guarda en archivos de texto. Puedes revisar estos archivos directamente. La ubicación predeterminada para estos archivos es %USERPROFILE%\AppData\Local\Temp\, y los archivos suelen tener nombres como fusionlog-...txt.

Estas opciones te permitirán revisar detalladamente cómo se están resolviendo las referencias de ensamblados durante la ejecución de tu aplicación y pueden ayudarte a identificar y solucionar problemas relacionados con la resolución de ensamblados.

9. ¿Qué herramientas de configuración posee el Framework?

En el contexto de los ensamblados, se puede hablar de herramientas específicas para trabajar con ellos, como:

**IL Disassembler (ildasm.exe)**

* **Propósito:** Desensambla un ensamblado de Common Intermediate Language (CIL) en código fuente equivalente.
* **Uso:** ildasm.exe NombreDelEnsamblado.dll.

**IL Assembler (ilasm.exe)**

* **Propósito:** Ensambla código fuente de Common Intermediate Language (CIL) en un ensamblado.
* **Uso:** ilasm.exe NombreDelArchivo.il.

**Assembly Linker (al.exe)**

* **Propósito:** Crea un ensamblado a partir de módulos.
* **Uso:** al.exe /out:NombreDelEnsamblado.dll Modulo1.netmodule Modulo2.netmodule.

**GACUtil (gacutil.exe)**

* **Propósito:** Instala o desinstala ensamblados en el Global Assembly Cache (GAC).
* **Uso:** gacutil.exe /i NombreDelEnsamblado.dll para instalar.

**SN (sn.exe)**

* **Propósito:** Firma ensamblados con claves fuertes y gestiona pares de claves.
* **Uso:** sn.exe -k clave.snk para generar un nuevo par de claves.

**ILMerge**

* **Propósito:** Combina múltiples ensamblados en uno solo.
* **Uso:** ilmerge /out:Resultado.dll Ensamblado1.dll Ensamblado2.dll.

Estas herramientas están más centradas en la manipulación directa de ensamblados, como desensamblar, ensamblar, firmar y fusionar, y son parte del conjunto de herramientas proporcionadas con el SDK de .NET Framework. Mientras que las herramientas de configuración y archivos de configuración que mencioné anteriormente están destinados a la configuración y personalización de aplicaciones durante el desarrollo y la ejecución.

10. ¿Qué herramientas del Framework se pueden utilizar desde el símbolo del sistema?

* **CSC.exe (C# Compiler):** Compila archivos fuente de C# en ensamblados ejecutables (.exe) o bibliotecas de clases (.dll).
* **MSBuild.exe:** Se utiliza para construir proyectos y soluciones .NET, lo que implica la compilación de ensamblados.
* **Ngen.exe:** Realiza la generación de imágenes nativas para ensamblados, lo que mejora el rendimiento al reducir el tiempo de carga.
* **Gacutil.exe:** Instala, desinstala y enumera ensamblados en el Global Assembly Cache (GAC), que es un repositorio centralizado de ensamblados compartidos.
* **Regasm.exe:** Se utiliza para registrar ensamblados COM (Component Object Model).
* **Sn.exe:** Administra claves y pares de claves pública/privada para la firma de ensamblados.
* **Editbin.exe:** Modifica el encabezado binario de un archivo ejecutable, lo cual puede ser relevante al trabajar con ensamblados.
* **ILasm.exe (IL Assembler):** Convierte código IL (Intermediate Language) en un archivo ejecutable o en un archivo de biblioteca de clases.

11. ¿Qué es el VBC, cómo y para qué se utiliza? 🡨 Compilador de Visual Basic

11. ¿Qué es el CSC, cómo y para qué se utiliza?

El CSC (C# Compiler) es el compilador de C# en el Framework .NET, representado por el ejecutable csc.exe. Se utiliza para convertir archivos fuente escritos en C# en ejecutables (.exe) o bibliotecas de clases (.dll). Al ejecutar csc.exe desde la línea de comandos, se compilan los archivos fuente, generando los archivos binarios correspondientes que pueden ejecutarse en el entorno .NET. Este compilador es esencial en el desarrollo de aplicaciones en C# y permite a los desarrolladores convertir su código fuente en ejecutables y bibliotecas de clases que son compatibles con la plataforma .NET.

12. ¿Para qué se utiliza el vinculador de ensamblado?

El vinculador de ensamblado, también conocido como linker de ensamblado, se utiliza para realizar la vinculación y la resolución final de referencias entre ensamblados en el proceso de compilación y construcción de una aplicación .NET. Aunque en la plataforma .NET, la vinculación es principalmente realizada en tiempo de ejecución, hay situaciones en las que se utiliza un vinculador de ensamblado en tiempo de compilación. Aquí hay algunas funciones y propósitos del vinculador de ensamblado:

* **Resolución de Referencias:** El vinculador de ensamblado resuelve las referencias entre ensamblados al identificar y enlazar las clases, métodos y recursos utilizados en un ensamblado con sus definiciones correspondientes en otros ensamblados.
* **Generación de Imágenes Nativas:** En algunos casos, el vinculador de ensamblado puede generar imágenes nativas, combinando y optimizando múltiples ensamblados en un solo archivo ejecutable nativo, lo que puede mejorar el rendimiento de la aplicación al reducir el tiempo de carga.
* **Eliminación de Código No Utilizado:** Puede eliminar el código no utilizado o innecesario, reduciendo así el tamaño del ensamblado final.
* **Manejo de Recursos y Metadatos:** El vinculador de ensamblado puede gestionar la inclusión de recursos y metadatos específicos del ensamblado, asegurando que estén disponibles y correctamente referenciados en el ensamblado final.

Es importante destacar que, en la plataforma .NET, gran parte de la vinculación y resolución de referencias se realiza dinámicamente en tiempo de ejecución a través del Common Language Runtime (CLR). Sin embargo, el uso de herramientas de vinculación de ensamblado en tiempo de compilación puede ser beneficioso en ciertos escenarios, como la generación de imágenes nativas y la optimización del tamaño del ensamblado.

13. ¿Para qué sirve la utilidad ILDASM?

La utilidad ILDASM (Intermediate Language Disassembler) es una herramienta proporcionada por el SDK de .NET que permite desensamblar (desmontar) un ensamblado compilado en código intermedio (IL, Intermediate Language) en lugar de en código fuente. ILDASM ofrece una vista detallada del código IL, metadatos, recursos y otros elementos del ensamblado. Aquí algunos propósitos y usos de ILDASM:

* **Inspección de Código IL:** ILDASM permite a los desarrolladores inspeccionar el código intermedio generado por el compilador .NET. Esto puede ser útil para entender cómo se traduce el código fuente a IL, especialmente cuando se trabaja con ensamblados de terceros o para propósitos de depuración avanzada.
* **Exploración de Metadatos:** Proporciona una vista detallada de los metadatos del ensamblado, incluidos detalles sobre tipos, miembros, referencias a otros ensamblados, y otros elementos relacionados con la estructura del programa.
* **Revisión de Recursos:** Permite examinar los recursos embebidos dentro del ensamblado, como imágenes, archivos de configuración, o cualquier otro recurso definido durante el desarrollo.
* **Generación de Archivos de Texto:** ILDASM puede generar archivos de texto que representan el código IL del ensamblado, lo que facilita su análisis o revisión. Estos archivos pueden ser examinados más fácilmente que el código binario.
* **Exportación de Tipos y Miembros:** Puede exportar tipos y miembros específicos del ensamblado para su revisión o análisis detallado.

La utilidad ILDASM se ejecuta desde la línea de comandos y proporciona una interfaz gráfica básica para navegar a través de la estructura del ensamblado. Aunque ILDASM es una herramienta poderosa para inspeccionar y entender ensamblados, generalmente se utiliza con fines de análisis y no para modificar el código o los metadatos del ensamblado.

14. ¿Para qué se utiliza el generador de imágenes nativas?

El generador de imágenes nativas, representado por la herramienta Ngen.exe (Native Image Generator), se utiliza en la plataforma .NET con el propósito de mejorar el rendimiento de las aplicaciones al precompilar el código intermedio (IL, Intermediate Language) en código nativo específico de la arquitectura del procesador. Aquí hay algunas razones para utilizar el generador de imágenes nativas:

* **Reducción del Tiempo de Carga:** Al generar imágenes nativas, se reduce el tiempo de carga de la aplicación, ya que el código nativo precompilado puede cargarse más rápido que el código IL, evitando así la compilación Just-In-Time (JIT) durante la ejecución.
* **Mejora del Rendimiento:** Al ejecutar código nativo en lugar de código IL, se puede lograr un rendimiento mejorado en algunas aplicaciones. Esto es especialmente relevante para aplicaciones que se ejecutan de manera repetida o en sistemas con recursos limitados.
* **Optimización para una Plataforma Específica:** Las imágenes nativas generadas son específicas de la arquitectura del procesador y del sistema operativo donde se realiza la compilación. Esto puede resultar en una optimización adicional para la plataforma de destino.
* **Menor Sobrecarga durante la Ejecución:** Al generar imágenes nativas durante el proceso de instalación o implementación, se reduce la sobrecarga asociada con la compilación JIT en tiempo de ejecución, ya que parte del trabajo de compilación se realiza de antemano.

Es importante tener en cuenta que el uso del generador de imágenes nativas no es siempre beneficioso en todos los escenarios. La elección de utilizar Ngen.exe debe basarse en el análisis del perfil de rendimiento específico de la aplicación. En algunos casos, el JIT puede ser más eficiente, especialmente en situaciones donde la aplicación tiene un inicio rápido y se ejecuta en un entorno diverso. Sin embargo, en casos de aplicaciones de servidor o aplicaciones críticas de rendimiento, la generación de imágenes nativas puede ser una estrategia válida para mejorar el rendimiento.

15. ¿Qué contiene el archivo AssemblyInfo?

El archivo AssemblyInfo.cs es un archivo de configuración en proyectos de C# que contiene información metadata sobre un ensamblado (assembly). Esta información se utiliza para proporcionar detalles adicionales sobre la biblioteca o aplicación compilada. Algunos de los elementos clave que puede contener AssemblyInfo.cs incluyen:

**Información General**

* **Título (Title):** El título del ensamblado.
* **Descripción (Description):** Una descripción breve del ensamblado.
* **Compañía (Company):** El nombre de la compañía o desarrollador.
* **Producto (Product):** El nombre del producto.

**Versión**

* **Versión (Version):** La versión del ensamblado, que consta de cuatro números: Major, Minor, Build y Revision.
* **Cultura (Culture):** La cultura asociada con el ensamblado, que puede afectar a cómo se manejan ciertos aspectos, como el formato de números y fechas.

**Configuración de Compatibilidad**

* **Atributo ComVisible:** Indica si los tipos en el ensamblado son visibles para componentes COM.
* **Atributo CLSCompliant:** Indica si el ensamblado cumple con las reglas de especificación del Lenguaje Común (Common Language Specification, CLS).

**Información del Autor**

* **Autor (Author):** El nombre del autor del ensamblado.

**Opciones de Ensamblado**

* **Atributo AssemblyVersion:** Define la versión del ensamblado.
* **Atributo AssemblyFileVersion:** Define la versión del archivo del ensamblado.
* **Atributo AssemblyInformationalVersion:** Puede contener información adicional, como la versión para mostrar en interfaces de usuario.

**Opciones de Compatibilidad con COM**

* **Atributo Guid:** Un identificador único global (GUID) que se utiliza para la interoperabilidad con componentes COM.

Estos son solo algunos ejemplos comunes. La información en AssemblyInfo.cs es utilizada por el compilador y el sistema de construcción para generar los metadatos del ensamblado. Personalizar este archivo proporciona detalles específicos sobre la biblioteca o aplicación que pueden ser útiles para herramientas, sistemas de compilación y para aquellos que consumen el ensamblado.

16. ¿Para qué sirve la clase AppDomain?

La clase AppDomain en .NET representa un dominio de aplicación, que es una unidad de aislamiento y seguridad en la que se ejecuta el código gestionado. Cada aplicación .NET se ejecuta en al menos un dominio de aplicación, y la clase AppDomain proporciona un entorno separado para la ejecución de aplicaciones. Aquí hay algunas funciones clave de la clase AppDomain:

* **Aislamiento de Aplicaciones:** AppDomain proporciona un nivel de aislamiento entre aplicaciones en ejecución. Esto significa que los recursos y los objetos en un dominio de aplicación no pueden afectar directamente a los recursos en otro dominio de aplicación.
* **Carga y Descarga Dinámica de Ensamblados:** Los dominios de aplicación permiten la carga y descarga dinámica de ensamblados durante la ejecución de una aplicación. Puedes crear un nuevo dominio de aplicación para cargar y ejecutar ensamblados sin afectar el dominio principal de la aplicación.
* **Aislamiento de Errores:** Si un error ocurre en un dominio de aplicación, generalmente no afecta otros dominios de la aplicación. Esto mejora la robustez de la aplicación, ya que un fallo en un dominio de aplicación no afecta directamente a otros dominios.
* **Gestión de Recursos:** Cada dominio de aplicación tiene su propio conjunto de recursos, incluidos tipos, ensamblados y otros objetos. Esto facilita la gestión y organización de recursos en una aplicación.
* **Seguridad:** Puedes establecer políticas de seguridad a nivel de dominio de aplicación, lo que permite controlar el acceso a recursos y servicios en cada dominio.
* **Recopilación de Basura Independiente:** La recopilación de basura (garbage collection) se realiza de forma independiente en cada dominio de aplicación. Esto significa que los objetos sin referencias solo se recopilan en el dominio específico, evitando la necesidad de una recopilación de basura global que podría afectar a toda la aplicación.

Ejemplo de creación de un nuevo AppDomain:

using System;

class Program

{

static void Main()

{

// Crear un nuevo AppDomain

AppDomain newDomain = AppDomain.CreateDomain("NuevoDominio");

// Realizar operaciones en el nuevo dominio...

// Descargar el dominio cuando ya no sea necesario

AppDomain.Unload(newDomain);

}

}

En este ejemplo, se crea un nuevo dominio de aplicación llamado "NuevoDominio". Puedes realizar operaciones en este nuevo dominio de aplicación, y cuando ya no sea necesario, puedes descargar el dominio utilizando AppDomain.Unload. Esto es útil, por ejemplo, cuando trabajas con ensamblados que quieres cargar y descargar dinámicamente.

UNIDAD II

17. ¿Qué entiende por arquitectura de un sistema de información?

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

lOMoARcPSD|6429649

4

18. ¿Cuál es la importancia de la arquitectura en el sistema de información?

19. ¿Cuáles son las metas de una arquitectura?

20. ¿Qué es una arquitectura clásica?

21. Defina y describa las capas de un modelo de arquitectura de N-capas

22. ¿Qué debe cumplir una arquitectura?

23. Enumere diferentes estrategias de arquitecturas según la filosofía de trabajo

24. Enumere y describa diferentes patrones que puede colaborar en el diseño de una

arquitectura

25. ¿Qué es SOA?

26. ¿Cuál es la diferencia entre capa y nivel en una arquitectura?

27. ¿Qué define una arquitectura?

28. ¿Qué ocurre si fallamos en la arquitectura?

29. ¿Qué permite una buena arquitectura?

30. ¿Es mejor una arquitectura de 20 capas que una de 4 capas?

31. ¿qué permite Una buena arquitectura?

UNIDAD III

32. ¿Qué es ADO.NET?

33. ¿Qué objetos componen el modelo de ADO.NET?

34. ¿Cuáles son las particularidades del modelo conectado?

35. ¿Qué objeto se debe utilizar para establecer una conexión a un origen de datos?

36. ¿Para qué se utiliza y qué contiene una cadena de conexión?

37. ¿Cómo se realiza la apertura y cierre de una conexión a una base de datos?

38. ¿Cómo se consulta el estado de una conexión y cuáles pueden ser estos?

39. ¿Cuáles son las propiedades, métodos y eventos más importantes del objeto

Conection?

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

lOMoARcPSD|6429649

5

40. ¿Cómo se inicia una transacción dentro de una conexión?

41. ¿Qué objeto representa a la transacción?

42. ¿Qué objeto se utiliza para invocar procedimientos almacenados u embeber SQL?

43. ¿Qué se debe configurar en un objeto Command para que pueda funcionar?

44. ¿Cuáles son las propiedades, métodos y eventos más importantes de la clase

Command?

45. ¿Cómo se pasan los parámetros que necesita un procedimiento almacenado para

poder funcionar?

46. ¿Para qué se utiliza el objeto DataReader?

47. ¿Cuáles son las propiedades, métodos y eventos más importantes de la clase

DataReader?

48. ¿En qué NameSpace se encuentran las clases para realizar conexiones, ejecutar

comandos o leer datos en el modelo conectado?

49. ¿Cuáles son las particularidades del modelo desconectado de ADO.NET?

50. ¿Qué representa un DataSet y para qué se usa?

51. ¿Qué colecciones maneja un DataSet?

52. ¿Cuáles son las propiedades, métodos y eventos más importantes de la clase DataSet?

53. ¿Qué objeto representa a una tabla de la base de datos?

54. ¿Cuáles son las propiedades, métodos y eventos más importantes de la clase

DataTable?

55. ¿Qué objeto representa a un registro de una tabla de la base de datos?

56. ¿Cuáles son las propiedades, métodos y eventos más importantes de la clase

DataRow?

57. ¿Qué objeto representa a una columna de una tabla de la base de datos?

58. ¿Cuáles son las propiedades, métodos y eventos más importantes de la clase

DataColumn?

59. ¿Qué objeto representa a una relación entre tablas de la base de datos?

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

lOMoARcPSD|6429649

6

60. ¿Cuáles son las propiedades, métodos y eventos más importantes de la clase

DataRelation?

61. ¿Para qué se utiliza el objeto DataAdapter?

62. ¿Cuáles son las propiedades, métodos y eventos más importantes de la clase

DataAdapter?

63. ¿Para qué se utiliza el objeto CommandBuilder?

64. ¿Cuáles son las propiedades, métodos y eventos más importantes de la clase

CommandBuilder?

65. ¿Qué tendría en cuenta para evitar los conflictos de concurrencia en las

actualizaciones?

66. ¿Qué ventajas posee utilizar ADO.NET y XML?

67. ¿Cómo es la estructura de un documento XML?

68. ¿Cómo se puede escribir y leer un documento XML?

69. ¿Para qué se utiliza un objeto del tipo XmlTextReader?

70. ¿Cuáles son las propiedades, métodos y eventos más importantes de la clase

XmlTextReader?

71. ¿Para qué se utiliza un objeto del tipo XmlTextWriter?

72. ¿Cuáles son las propiedades, métodos y eventos más importantes de la clase

XmlTextWriter?

73. ¿Qué es un XMLNode?

74. ¿Para qué se usa un XMLNode?

75. ¿Qué es un XMLDocument?

76. ¿Para qué se usa un XMLDocument?

77. ¿Qué es un XSLTransform?

78. ¿Para qué se usa un XSLTransform?

79. ¿Cómo puede realizar la Lectura del esquema de datos XML?

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

lOMoARcPSD|6429649

7

UNIDAD IV

80. ¿Qué es un control de usuario personalizado?

81. ¿Enumere al menos tres formas de aprovechar los controles de usuarios

personalizados?

82. ¿Qué elemento puede contener un control de usuario personalizado?

83. ¿Qué ventajas posee utilizar controles de usuarios personalizados?

84. ¿Cómo puedo mejorar la interfaz de un control de usuario personalizado?

85. ¿Qué tipo de proyectos debo generar para lograr un control de usuario personalizado?

86. ¿Para qué se puede utilizar el componente de licencia?

87. ¿Cómo puede un control de usuario conceder licencias en tiempo de diseño?

88. ¿Cómo puede un control de usuario conceder licencias en tiempo de ejecución?

89. ¿Cómo puedo alojar un control de usuario en un navegador?

90. ¿Qué es una expresión regular?

91. ¿Cómo se clasifican los caracteres que componen el lenguaje de las expresiones

regulares?

92. ¿Qué objeto se utiliza para trabajar con expresiones regulares?

93. ¿Qué colección representa las coincidencias encontradas entre el texto de origen i el

pattern?

94. ¿Qué objeto representa una ocurrencia encontrada en el texto de origen respeto del

pattern?

95. ¿Para qué se utiliza la clase Group?

96. ¿Para qué se utiliza la clase CaptureCollection?

97. ¿Para qué se utiliza la clase CaptureCapture?

98. ¿Cuáles son las propiedades, métodos y eventos más importantes de la clase RegEx?

99. ¿Cuáles son las propiedades, métodos y eventos más importantes de

MatchCollection?

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

lOMoARcPSD|6429649

8

100. ¿Cuáles son las propiedades, métodos y eventos más importantes de la clase

Match?

101. ¿Cuáles son las propiedades, métodos y eventos más importantes de la clase

Group?

102. ¿Cuáles son las propiedades, métodos y eventos más importantes de la clase

CaptureCollection?

103. ¿Cuáles son las propiedades, métodos y eventos más importantes de la clase

Capture?

104. ¿Cuáles son los componentes que se usan en un reporte?

105. ¿Cuáles son las partes del reporte?

106. ¿Cómo vincular el resultado obtenido de la Base de datos con el reporte?

107. ¿Cuáles son las variantes que el IDE permite incorporar para generar reportes?

108. ¿Hay alternativas de reportes con la base de datos?

UNIDAD V

109. ¿Cuáles son los objetos gráficos de GDI 2D más importantes? describa para

que sirve cada uno.

110. ¿Para qué se usa el objeto Pen?

111. ¿Cómo se rellena una forma?

112. ¿Qué se utiliza para generar pinceles?

113. ¿Qué son y para qué sirve generar regiones?

114. ¿Qué son y para qué se utilizan las regiones?

115. ¿Qué elementos utilizaría para tratar una imagen?

116. ¿Cómo construye y administra un mapa de bits transparente o semitransparente?

117. ¿Cómo aprovecharía y utilizaría íconos desde los objetos gráficos?

118. ¿Cuáles son los objetos más importantes para tratar la tipografía con los métodos

gráficos?

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

lOMoARcPSD|6429649

9

119. ¿Qué efectos se le pueden aplicar a la tipografía? Describa cada uno de ellos

UNIDAD VI

120. ¿Cuál es la importancia de la recursividad en la estratetgia de diseño de algoritmos?

121. ¿Cuáles son los tipos de recursividad?

122. ¿Explque la tecnica de dividir y conquistar?

123. ¿Qué entiende por programacion dinámica?

124. ¿Qué es un algorotmo Ávido?

125. ¿Dado un algoritmo y dos implementaciones suyas I1 e I2, que tardan T1(n) y T2(n)

respectivamente que afirma el proceso de invarianza?

126. ¿Qué tiene mas infliencia en el proceso de comparar la eficia temporalde dos

algoritmos?

127. ¿Qué se usa para medir el tiempo de ejecucion de un algoritmo?

128. ¿enuemre las operaciones elementales?

129. ¿Explique que es una medida asintomática?

130. ¿Cómo aumenta el tiempo de ejecucion de un algoritmo en función del tamaño de

etrada?

131. ¿Cuál es la diferencia entre programacion paralela y secuencial?

132. ¿Dónde se utiliza un lagoritmo distruibuido?

133. ¿Qué es un dato de Test?

134. ¿Cuáles son las fases de la resolucion de problemas?

135. ¿Cómo se mide la eficiencia de un algoritmo?

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

lOMoARcPSD|6429649

10

GUIƵA DE EJERCICIOS

UNIDAD I

1. Desarrollar un programa que permita ejecutar el desensamblador de código

intermedio y que se pueda seleccionar desde un listado el archivo a desensamblar.

¿Qué observa?

2. Desarrollar un ensamblado que posea nombre seguro y quede registrado en el GAC.

Demostrar que ha quedado registrado.

3. Desarrollar un programa y compilarlo con el VS desde la línea de comandos.

UNIDAD II

4. Diagrame y modele una estrategia de capas para un sistema cliente servidor, sabiendo

que el cliente puede estar dispuesto en distintas plataformas

5. Desarrolle un modelo de capas para un sistema de gestión comercial de una empresa

transportadora de mercadería. Tener en cuenta la posibilidad que mucha de las

funciones puede ser utilizadas en otros sistemas.

6. ¿Cuál sería la arquitectura para integrar diferentes sistemas desarrolladores en

diferentes lenguajes y sistemas operativos

UNIDAD III

1. Desarrollar un programa que permita acceder a una tabla de una base de datos y

mostrarlos en una grilla.

2. Desarrollar un programa que permita acceder a una tabla de una base de datos y que

en la misma se puedan agregar, modificar y borrar registros.

3. Desarrollar un programa que permita acceder a una tabla de una base de datos y que

en la misma se puedan agregar, modificar, borrar y consultar registros utilizando

procedimientos almacenados.

4. Desarrollar un programa que permita acceder a dos tablas relacionadas de una base

de datos y que en la misma se puedan agregar, modificar, borrar y consultar registros

utilizando procedimientos almacenados. Para cada operación realizada abrir una

transacción y controlar la misma.

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

lOMoARcPSD|6429649

11

5. Desarrollar un programa que permita acceder a dos tablas relacionadas de una base

de datos aplicando el modelo desconectado. Levantar los datos a un DataSet. Agregar,

modificar, borrar y consultar registros. Luego actualizar la base de datos por medio

del adaptador.

6. Sobre el ejercicio anterior generar los SQL del adaptador con el objeto

CommandBuilder.

7. Desarrollar un programa aplicando todo lo aprendido sobre ADO para administrar un

video club.

8. Desarrollar un programa que permita acceder a los datos de una base de datos

aplicando el modelo desconectado. Levantar los datos a un DataSet. Agregar,

modificar, borrar y consultar registros. Agregar la posibilidad que los datos del

DataSet se guarden en un documento XML y que luego se puedan leer desde allí.

9. Desarrollar un programa que permita acceder a los datos de una base de datos, y que

las consultas se graben en un documento XML para luego ser visualizados desde un

navegador.

UNIDAD IV

10. Desarrolle un control de usuario que emule las propiedades de una caja de texto

tradicional pero que permita configurar las validaciones de sus entradas

11. Crear un control de usuario para Login

12. Desarrollar un control de usuario personalizado que herede de un control de usuario

existente. Utilizarlo en un programa de acceso a datos para validad algún ingreso.

13. Desarrollar un control de usuario personalizado desde cero (p.e. un botón redondo

con las funcionalidades similares a un botón de comando) y aplicarlo a un programa.

14. Desarrollar un control de usuario personalizado que agrupe varios controles de

usuario existentes (p.e. una botonera para un sistema de ABCM) y aplicarlo a un

programa.

15. Crear un reporte de clientes morosos

16. Crear un reporte personalizados para mostrar el libre deuda de un cliente

17. Desarrollar un programa utilizando expresiones regulares que permita validar códigos

de barra que responden a la norma EAN-13.

18. Desarrollar un programa para administrar un ABMC de alumnos donde los datos

capturados (p.e. mail, teléfono, nombre etc) se validen con expresiones regulares.

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

lOMoARcPSD|6429649

12

19. Desarrollar un programa que sea un probador de expresiones regulares de manera que

se pueda ingresar un texto de origen e ir escribiendo dinámicamente una expresión

regular. Este programa deberá informar sobre cuantas coincidencias se encuentran

mientras se escribe la expresión regular y cuáles son.

UNIDAD V

1. Desarrollar un programa que utilizando los métodos gráficos permita escribir un

nombre ingresado por el usuario.

2. Desarrollar un programa que utilizando los métodos gráficos dibuje una pirámide y

que los colores estén degradados desde el extremo superior de la pirámide hasta su

base. Estos colores deben generarse aleatoriamente y cambiar cada 100 milisegundos.

3. Desarrollar un programa que utilizando los métodos gráficos permita graficar

funciones matemáticas (p.e seno, coseno, tangente, X2 – 2X + 5 etc.)

4. Desarrollar un programa que utilizando los métodos gráficos permita aplicarle efectos

a las letras del texto que ingresa el usuario.

5. Desarrollar un programa que utilizando los métodos gráficos emule un tablero

electrónico donde se pueden proyectar leyendas. El mismo debe permitir configurar

el tamaño, color y otras características relevantes del tablero.

Unidad VI

6. Evalúe el tiempo de ejecucion del método burbuja

7. Analizar las operaciones elementales en un algoritmo que busca valor en un arreglo

ordenado

8. Calcular el tiempo de ejecución de la instrucción FOR

9. Analizar, calcular el costo y codificar los siguientes ejercicios:

a. Calcular la sumatorio de los n primeros multiplos de 3.

b. Hallar la multiplicacion de los medios (1/2) hasta 3

c. Calcular la suma de los primero 10 numneros impares

d. Calcular la división de un numero entero A con un número entero B por restas

sucesivas.

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

lOMoARcPSD|6429649

13

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

lOMoARcPSD|6429649

14

GUIƵA DE ABORDAJE BIBLIOGRAƵFICO

UNIDAD I

Deitel, Harvey M., Deitel, Paul J.(2007). Cómo programar en C#. (4 Ed. cap 1) Mexico:

Pearson Educación

Amplatoria: “Programación Avanzada Con Visual Basic 2005”. ”. Francesco Balena. Mc

Graw Hill. 2003. España. Capítulo 14.

UNIDAD II

Simon, Bennet. (2006). Análisis y Diseño orientado a objetos de sistemas. (3ed, cap 12)

Madrid: McGraw-Hill.

UNIDAD III

Deitel, Harvey M., Deitel, Paul J.(2007). Cómo programar en C#. (4 Ed. cap 19 y 20) Mexico:

Pearson Educación

Ampliatoria: “Programación Avanzada Con Visual Basic 2005”. Francesco Balena. Mc

Graw Hill. 2003. España. Capítulo 22.

UNIDAD IV

Microsoft . https://msdn.microsoft.com/es-es/library/hs600312(v=vs.110).aspx. (2017)

Microsoft . https://msdn.microsoft.com/es-es/library/hs600312(v=vs.110).aspx. (2017)

Microsfot. https://msdn.microsoft.com/es-es/library/ms252073.aspx. (2017)

Microsfot. https://msdn.microsoft.com/en-us/library/ms252104.aspx (2017)

Ampliatoria: “Programación Avanzada con Microsoft Visual Basic .NET”. Francesco

Balena. Mc Graw Hill. 2003. España. Capítulo 25.

UNIDAD V

Deitel, Harvey M., Deitel, Paul J.(2007). Cómo programar en C#. (4 Ed. cap 15 y 17) Mexico:

Pearson Educación

Unidad VI

Brassard, G.; Bratley, P.. Fundamentos de algoritmia.‐‐ Madrid : Prentice Hall, 1997. xiii, 579 páginas

Rosa Guerequeta, Antonio Vallecillo. (1998). Técnicas de Diseño de Algoritmos. Universidad de

malaga. (http://www.lcc.uma.es/~av/Libro/)

Maria, Gopmez fuetes. Jorge, Cervantes. (2014). Introducción al análisis y diseño de

algoritmos. Universidad autónoma metropolitana.

(http://www.cua.uam.mx/pdfs/conoce/libroselec/Notas\_Analisis\_AlgoritmosVF.pdf)

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

lOMoARcPSD|6429649

15

Descargado por Gerardo Tordoya (zherar7ordoya@outlook.com)

lOMoARcPSD|6429649