


GENERACIÓN DE INTERFACES DE USUARIO Y CREACIÓN DE COMPONENTES VISUALES

Contenido

2.1. Introducción a las herramientas de desarrollo.	2
2.1.1 Ciclo de vida del software.....	2
Planificación	2
Análisis	2
Diseño	3
Implementación.....	3
Pruebas.....	3
Instalación o despliegue	4
Uso y mantenimiento	4
2.1.2 Herramienta de Desarrollo Integrado (IDE)	5
Características de los entornos de desarrollo	5
Principales IDEs.....	6
2.2. Configuración del IDE: Plataforma Microsoft .NET.....	8
2.2 Visual Studio y .NET Framework	11
2.3 Soluciones y Proyectos en Visual Studio	13
2.4 Elementos de un proyecto.	¡Error! Marcador no definido.
2.5 Introducción a C#	¡Error! Marcador no definido.
2.6 Objetos, clases, propiedades y métodos	¡Error! Marcador no definido.
2.7 Desarrollo de interfaces gráficas para aplicaciones.	¡Error! Marcador no definido.
2.8 Edición del código generado por la herramienta de diseño. .	¡Error! Marcador no definido.
2.9 Aplicaciones de Windows Forms.....	¡Error! Marcador no definido.
2.10 Diálogos modales y no modales.	¡Error! Marcador no definido.
2.11 Generación de interface de acceso a ficheros de texto y XML.....	¡Error! Marcador no definido.
2.12 Enlace de componentes a orígenes de datos.	¡Error! Marcador no definido.
2.13 Microsoft Team Foundation Server: Sistema integrado para el control del código fuente y la gestión del ciclo de vida de las aplicaciones	¡Error! Marcador no definido.

	UT 2 - <i>Generación de interfaces de usuario y Creación de componentes visuales</i>	CFGS Desarrollo de Aplicaciones Multiplataforma Módulo: DDI
---	---	--

2.1. Introducción a las herramientas de desarrollo.

Una herramienta de desarrollo es aquella que sirve para agilizar cualquier parte del proceso de dentro del ciclo de vida del desarrollo de software.

Aunque existen diferentes ciclos de desarrollo de software, la normativa ISO/IEC/IEEE 12207:2017 establece:

“Un marco común para los procesos del ciclo de vida de los programas informáticos, con una terminología bien definida, a la que pueda remitirse la industria del software. Contiene procesos, actividades y tareas aplicables durante la adquisición, el suministro, el desarrollo, el funcionamiento, el mantenimiento o la eliminación de sistemas, productos y servicios informáticos. Estos procesos del ciclo de vida se llevan a cabo mediante la participación de los interesados, con el objetivo final de lograr la satisfacción del cliente”.

2.1.1 Ciclo de vida del software

Las etapas del desarrollo de software más aceptadas son:


Planificación

Antes de empezar un proyecto de desarrollo de un sistema de información, es necesario hacer ciertas tareas que influirán decisivamente en el éxito del mismo.

Algunas de las tareas de esta fase incluyen actividades como la determinación del ámbito del proyecto, la realización de un estudio de viabilidad, el análisis de los riesgos asociados, la estimación del coste del proyecto, su planificación temporal y la asignación de recursos a las diferentes etapas del proyecto.

Análisis

La etapa de análisis en el ciclo de vida del software corresponde al proceso a través del cual se intenta descubrir qué es lo que realmente se necesita y se llega a una comprensión adecuada de los requerimientos del sistema (las características que el sistema debe poseer).

	UT 2 - <i>Generación de interfaces de usuario y Creación de componentes visuales</i>	CFGS Desarrollo de Aplicaciones Multiplataforma Módulo: DDI
---	---	--

Diseño

En esta fase se estudian posibles opciones de implementación para el software que hay que construir, así como decidir la estructura general del mismo. El diseño es una etapa compleja y su proceso debe realizarse de manera iterativa.

Es posible que la solución inicial no sea la más adecuada, por lo que en tal caso hay que refinarla. No obstante, hay catálogos de patrones de diseño muy útiles que recogen errores que otros han cometido para no caer en la misma trampa.

Implementación

En esta fase hay que elegir las herramientas adecuadas, un entorno de desarrollo que facilite el trabajo y un lenguaje de programación apropiado para el tipo de software a construir. Esta elección dependerá tanto de las decisiones de diseño tomadas como del entorno en el que el software deba funcionar.


Al programar, hay que intentar que el código no sea indescifrable siguiendo distintas pautas como las siguientes:

- Identificar correctamente las variables y su alcance.
- Elegir algoritmos y estructuras de datos adecuadas para el problema.
- Mantener la lógica de la aplicación lo más sencilla posible.
- Documentar y comentar adecuadamente el código de los programas.
- Facilitar la interpretación visual del código utilizando reglas de formato de código previamente consensuadas en el equipo de desarrollo.

También hay que tener en cuenta la adquisición de recursos necesarios para que el software funcione, además de desarrollar casos de prueba para comprobar el funcionamiento del mismo según se vaya programando.

Pruebas

La fase de pruebas del ciclo de vida del software busca detectar los fallos cometidos en las etapas anteriores para corregirlos. Por supuesto, lo ideal es hacerlo antes de que el usuario final se los encuentre. Se dice que una prueba es un éxito si se detecta algún error. El departamento QA (Quality Assurance) algo así como departamento de “garantía de calidad” diseña y define todos los parámetros de aceptación de un paquete de software y ejecuta las pruebas funcionales, unitarias, de integración y las pruebas de regresión, es decir, efectúa las pruebas a nivel de código fuente (conocidas como White

	UT 2 - <i>Generación de interfaces de usuario y Creación de componentes visuales</i>	CFGS Desarrollo de Aplicaciones Multiplataforma Módulo: DDI
---	---	---

Box Testing).

El departamento QA está presente desde la fase de diseño del producto asegura que todos los desarrolladores siguen los mismos estándares de calidad, dentro de una gran corporación. Trabaja paralelamente con el departamento de desarrollo, jefes de proyecto y el cliente.

Instalación o despliegue

Es poner el software en funcionamiento, por lo que hay que planificar el entorno teniendo en cuenta las dependencias existentes entre los diferentes componentes del mismo.

Es posible que haya componentes que funcionen correctamente por separado, pero que al combinarlos provoquen problemas. Por ello, hay que usar combinaciones conocidas que no causen problemas de compatibilidad.

La planificación de un despliegue es un punto de fricción entre diferentes departamentos dentro de una organización. Mientras el departamento de producto intentará que el despliegue se lleve a cabo lo antes posible para contentar al cliente el departamento de desarrollo tiene que valorar aspectos paralelos que pueden afectar a la instalación (p.e. parada de servidores que pueden alojar otras funcionalidades).


Uso y mantenimiento

Esta es una de las fases más importantes del ciclo de vida de desarrollo del software. Cuyos objetivos son:

- Eliminar los defectos detectados durante su vida útil (mantenimiento correctivo).
- Adaptarlo a nuevas necesidades (mantenimiento adaptativo).
- Añadirle nuevas funcionalidades (mantenimiento perfectivo).

Son varios los factores influyen en el tiempo que hay que invertir en el mantenimiento de una aplicación como un mal diseño, deficiente documentación, inestabilidad del equipo de desarrollo y el número de usuarios que lo utilicen. Cuantos más usuarios utilicen una aplicación más tiempo hay que invertir en su mantenimiento. La principal razón es que se usará más (incluso de formas que no se habían previsto) y habrá más propuestas de mejoras.

Es una de las etapas que hay que optimizar para que la rentabilidad de un software sea óptima. Un mayor tiempo de mantenimiento repercute directamente en los costes del producto, pero tampoco se debe escatimar recursos en esta fase. Un mal mantenimiento

	UT 2 - <i>Generación de interfaces de usuario y Creación de componentes visuales</i>	CFGS Desarrollo de Aplicaciones Multiplataforma Módulo: DDI
---	---	--

repercute en la imagen que se transmite al cliente final, una mala imagen puede empujarle a migrar a otra aplicación o que este no recomiende la aplicación a otros potenciales clientes.

2.1.2 Herramienta de Desarrollo Integrado (IDE)

Una herramienta de desarrollo integrado (IDE) es un software empleado para desarrollar otro tipo de software. IDE es el acrónimo del término inglés Integrated Development Environment. El objetivo de una IDE es agilizar todo el proceso de diseño de software, ofreciendo un servicio integral al programador.


La mayoría de las herramientas IDE ofrecen un entorno amigable que permite trabajar con diferentes lenguajes de programación y distintos sistemas operativos, aunque hayan sido diseñados para ser empleados específicamente en uno de ellos.

Un IDE es imprescindible tanto en el ámbito del en el Desarrollo de Aplicaciones Multiplataforma (DAM) como Desarrollo de Aplicaciones Web (DAW). Hace que la tarea del programador sea más sencilla gracias a las herramientas que tiene incorporadas, como compiladores, depuradores o bibliotecas, y esto se traduce en un aumento de la productividad.

Características de los entornos de desarrollo

Cualquier IDE debe tener una serie de características básicas que garanticen que la experiencia del usuario será satisfactoria. Todo IDE debe contar con:

- Editor de código. Se trata de un editor de texto creado exclusivamente para trabajar con el código fuente de programas informáticos.
- Compilador. Un programa encargado de traducir las instrucciones en código fuente, escritas en lenguaje de programación, a código objeto, el único lenguaje que el ordenador entiende.
- Depurador o debugger. Un programa que permite probar y buscar errores en otros programas.

	UT 2 - <i>Generación de interfaces de usuario y Creación de componentes visuales</i>	CFGS Desarrollo de Aplicaciones Multiplataforma Módulo: DDI
---	---	--

- Linker. Es la herramienta con la que combinar diferentes archivos de código fuente para convertirlos en un único fichero ejecutable.
- Refactorización de código. Proceso en el que se recurre a funciones como el reformato o la encapsulación para mejorar el código fuente.

Principales IDEs

El abanico de IDEs es muy amplio. Decantarse por uno u otro dependerá básicamente de las exigencias y necesidades de cada programador, que puede utilizar IDEs diferentes para trabajos distintos. Entre las alternativas más utilizadas y mejor valoradas están:

Eclipse

Eclipse es, probablemente, uno de los IDEs más utilizados y la clave está en que se trata de un entorno de desarrollo integrado de código abierto y multiplataforma. Desarrollado por IBM en su inicio, hoy lo gestiona la Fundación Eclipse, una entidad legal sin ánimo de lucro. Cada año cuenta con una versión actualizada que incluye una enorme biblioteca de plugins que permiten desarrollar todo tipo de aplicaciones, empleando Java, JSP, C, C++, Python, Ruby, PHP...


Cuenta con una lista de tareas y un editor de texto que muestra el contenido del fichero en el que se trabaja, la compilación se lleva a cabo en tiempo real y, a medida que se va avanzando en el diseño, su asistente propone una serie de recomendaciones para solucionar errores y optimizar códigos.

NetBeans

NetBeans es otro IDE de código abierto y gratuito, con el que crear aplicaciones empleando lenguajes como Java, PHP, C ++, HTML... Puede ejecutarse en cualquier sistema operativo y entre sus ventajas está que permite programar en Framework de Java Swing, lo que facilita el desarrollo de aplicaciones con entorno gráfico, es decir, mucho más dinámicas. También puede programar en Android, instalando los plugins necesarios. Entre sus atractivos está el manejo automático de la memoria y una interfaz de usuario muy cómoda.

Visual Studio

Visual Studio es la apuesta de Microsoft, un IDE que ofrece al programador múltiples

	UT 2 - <i>Generación de interfaces de usuario y Creación de componentes visuales</i>	CFGS Desarrollo de Aplicaciones Multiplataforma Módulo: DDI
---	---	--

funciones para crear códigos, depurar errores o realizar pruebas en el desarrollo de aplicaciones con el marco .NET y en cualquier plataforma. Su editor de código es compatible con IntelliSense (función de autocompletado) y su depurador funciona tanto a nivel fuente como a nivel máquina. Cuenta con un generador de perfiles de código y permite crear aplicaciones GUI (Graphical User Interface), diseños web o incluso ofrece la posibilidad de utilizar su diseñador de esquemas de base de datos.

Xcode

Xcode es el IDE oficial de Apple, creado para desarrolladores de Mac e iOS que también facilita programar en Java. Incluye infinidad de herramientas para desarrollar software para iOS, MacOS, watchOS y tvOS. Incorpora un depurador, un generador de GUI y permite el autocompletado de perfiles. También ofrece soporte para AppleScript, Python, Ruby, Swift C, C ++, Objective-C y Objective-C ++ y C#.

IntelliJ Idea

IntelliJ Idea fue creado por JetBrains, creadores también de Kotlin. Permite utilizar diferentes lenguajes de programación y trabajar con distintas versiones de software sin que afecte al desarrollo del trabajo


BlueJ

BlueJ es otro de los IDEs multiplataforma para el lenguaje de programación Java y fue creado como herramienta de apoyo a la enseñanza, aunque también hace posible desarrollar software a pequeña escala. Todas sus características facilitan la labor de aprender programación orientada a objetos.

La elección del IDE condicionará el resto del proyecto. Su interfaz debe ser atractiva y sencilla. A las tres funciones clave que todo IDE debe incorporar (editor de texto, compilador y depurador) se añaden otras alternativas, como las herramientas de integración e implementación continuas.

Crecimiento uso IDE en el mundo: <https://pypl.github.io/IDE.html>

Puedes ampliar información sobre diferentes IDEs comerciales en <https://www.guru99.com/web-development-ide.html>

	UT 2 - <i>Generación de interfaces de usuario y Creación de componentes visuales</i>	CFGS Desarrollo de Aplicaciones Multiplataforma Módulo: DDI
---	--	--

2.2. Configuración del IDE: Plataforma Microsoft .NET

Microsoft .NET es una plataforma que agrupa diferentes plataformas de software, originalmente desarrollado como una competencia directa a la plataforma Java.

.NET es una plataforma de código abierto gratuita y multilenguaje. Se puede descargar desde <https://dotnet.microsoft.com/download/dotnet>

Una aplicación de .NET se desarrolla y se ejecuta en una o varias *implementaciones de .NET*. Las implementaciones de .NET incluyen .NET Framework, .NET 5 (y .NET Core) y Mono. Hay una especificación de API común a varias implementaciones de .NET que se denomina .NET Standard.


La plataforma .NET proporciona herramientas y tecnologías para crear aplicaciones que aprovechen las capacidades de múltiples plataformas: Windows, Android, iOS o Linux entre otras.

.NET Framework es la implementación de .NET original que existe desde 2002. Contiene API específicas de Windows adicionales, como API para el desarrollo de escritorio de Windows con Windows Forms y WPF. .NET Framework está optimizado para crear aplicaciones de escritorio de Windows.

Para desarrollar un programa basado en el .NET Framework no se necesita mucho. Para escribir el programa simple es suficiente un **editor de texto** y un **compilador que se incluye en el framework** (responsable de traducir el código escrito del programa a un lenguaje que pueda ser entendido por la máquina en la que corre el programa. Con estos dos componentes ya se pueden crear aplicaciones sencillas.

El diseño de .NET Framework está enfocado a cumplir los objetivos siguientes:

- Proporcionar un entorno de programación orientada a objetos coherente en el que el código de los objetos se pueda almacenar y ejecutar de forma local, ejecutar de forma local pero distribuida en Internet o ejecutar de forma remota.
- Proporcionar un entorno de ejecución de código que:
 - Minimice los conflictos de implementación de software y control de versiones.
 - Fomente la ejecución segura de código, incluso del creado por terceros desconocidos o que no son de plena confianza.

	UT 2 - <i>Generación de interfaces de usuario y Creación de componentes visuales</i>	CFGS Desarrollo de Aplicaciones Multiplataforma Módulo: DDI
---	---	--

- Elimine los problemas de rendimiento de los entornos con scripts o interpretados.
- Ofrecer al desarrollador una experiencia coherente entre tipos de aplicaciones muy diferentes, como las basadas en Windows o en web.
- Basar toda la comunicación en estándares del sector para garantizar que el código basado en .NET Framework se integre con otro código.


.NET Framework consta de dos componentes principales: **Common Language Runtime (CLR)** y **la biblioteca de clases de .NET Framework**.

Common Language Runtime es el fundamento de .NET Framework. Es el motor que administra el código en tiempo de ejecución y proporciona servicios centrales, como la administración de memoria, la administración de subprocesos y la comunicación remota, al tiempo que aplica una seguridad de tipos estricta y otras formas de especificación del código que promueven su seguridad y solidez. De hecho, el concepto de administración de código es un principio básico del motor en tiempo de ejecución. El código destinado al tiempo de ejecución se denomina código administrado, a diferencia del resto de código, que se conoce como código no administrado.

La biblioteca de clases es una colección completa orientada a objetos que proporciona una colección de código probado y reutilizable al que pueden llamar los desarrolladores desde sus propias aplicaciones. Se puede emplear para desarrollar aplicaciones que abarcan desde las tradicionales herramientas de interfaz gráfica de usuario (GUI) o de línea de comandos hasta aplicaciones basadas en las innovaciones más recientes proporcionadas por ASP.NET, como formularios Web Forms y servicios web XML.

Los servicios que ofrece .NET Framework a las aplicaciones en ejecución son los siguientes:

- Administración de la memoria. En muchos lenguajes de programación, los programadores son responsables de asignar y liberar memoria y de administrar la vida útil de los objetos. En las aplicaciones de .NET Framework, CLR proporciona estos servicios en nombre de la aplicación.
- Sistema de tipos comunes. En los lenguajes de programación tradicionales, el compilador define los tipos básicos, lo que complica la interoperabilidad entre

	UT 2 - <i>Generación de interfaces de usuario y Creación de componentes visuales</i>	CFGS Desarrollo de Aplicaciones Multiplataforma Módulo: DDI
---	---	--

lenguajes. En .NET Framework, los tipos básicos los define el sistema de tipos de .NET Framework y son comunes a todos los lenguajes que tienen como destino .NET Framework.

- Biblioteca de clases extensa. En lugar de tener que escribir cantidades extensas de código para controlar operaciones comunes de programación de bajo nivel, los programadores usan una biblioteca de tipos accesible en todo momento y sus miembros desde la biblioteca de clases de .NET Framework.
- Marcos y tecnologías de desarrollo. .NET Framework incluye bibliotecas para determinadas áreas de desarrollo de aplicaciones, como ASP.NET para aplicaciones web, ADO.NET para el acceso a los datos, Windows Communication Foundation para las aplicaciones orientadas a servicios y Windows Presentation Foundation para las aplicaciones de escritorio de Windows.
- Interoperabilidad de lenguajes. Los compiladores de lenguajes cuya plataforma de destino es .NET Framework emiten un código intermedio denominado Lenguaje intermedio común (CIL), que, a su vez, se compila en tiempo de ejecución a través de Common Language Runtime. Con esta característica, las rutinas escritas en un lenguaje son accesibles para otros lenguajes, de modo que los programadores puedan centrarse en crear aplicaciones en su lenguaje preferido.
- Compatibilidad de versiones. Con raras excepciones, las aplicaciones que se desarrollan con una versión determinada de .NET Framework se ejecutan sin modificaciones en una versión posterior.
- Ejecución en paralelo. .NET Framework ayuda a resolver conflictos entre versiones y permite que varias versiones de Common Language Runtime coexistan en el mismo equipo. Esto significa que pueden coexistir varias versiones de las aplicaciones, y que una aplicación se puede ejecutar en la versión de .NET Framework con la que se compiló.


Para proyectos .NET Microsoft ofrece estos entornos de desarrollo integrado:

- [Visual Studio](#)

Solo se ejecuta en Windows. Dispone de una amplia funcionalidad integrada diseñada para trabajar con .NET. La edición Community es gratuita para estudiantes, colaboradores de código abierto y particulares.

- [Visual Studio Code](#)

Es compatible con Windows, macOS y Linux. De código abierto y gratuito. Hay

	UT 2 - <i>Generación de interfaces de usuario y Creación de componentes visuales</i>	CFGS Desarrollo de Aplicaciones Multiplataforma Módulo: DDI
---	---	--

extensiones disponibles para trabajar con lenguajes de .NET.

- [Visual Studio para Mac](#)

Solo se ejecuta en macOS. Para desarrollar aplicaciones y juegos de .NET para iOS, Android y la web.

- [GitHub Codespaces](#)

Un entorno de Visual Studio Code en línea, actualmente en versión beta.

2.2 Visual Studio y .NET Framework


Visual Studio es una herramienta de desarrollo integrado que permite desarrollar aplicaciones, en cualquier entorno que soporte la plataforma .NET Framework

El entorno de desarrollo integrado de Visual Studio es un panel de inicio que se puede usar para editar, depurar y compilar código y, después, publicar una aplicación. Más allá del editor estándar y el depurador que proporcionan la mayoría de IDE, Visual Studio incluye compiladores, herramientas de finalización de código, diseñadores gráficos y muchas más características para facilitar el proceso de desarrollo de software.

- Versiones de Visual Studio:
- Visual Studio 6.0: publicada en junio de 1997
- Visual Basic 6.0: publicada en junio de 1998
- Visual Studio 2003: publicada en abril de 2003
- Visual Studio 2005: publicada en noviembre de 2005
- Visual Studio 2008: publicada en noviembre de 2007
- Visual Studio 2010: publicada en abril de 2010
- Visual Studio 2012: publicada en septiembre de 2012
- Visual Studio 2013: publicada en octubre de 2013
- Visual Studio 2015: publicada en julio de 2015
- Visual Studio 2017: Publicada en mayo 2017
- Visual Studio 2019: Publicada en abril 2019
- Visual Studio 2022: Publicada en noviembre 2021

Existen tres ediciones de Visual Studio, la edición gratuita Community y las ediciones comerciales Professional y Enterprise.

Se puede descargar Visual Studio en <https://visualstudio.microsoft.com/es/>, su página

	<p>UT 2 - <i>Generación de interfaces de usuario y Creación de componentes visuales</i></p>	<p>CFGS Desarrollo de Aplicaciones Multiplataforma</p> <p>Módulo: DDI</p>
---	---	---

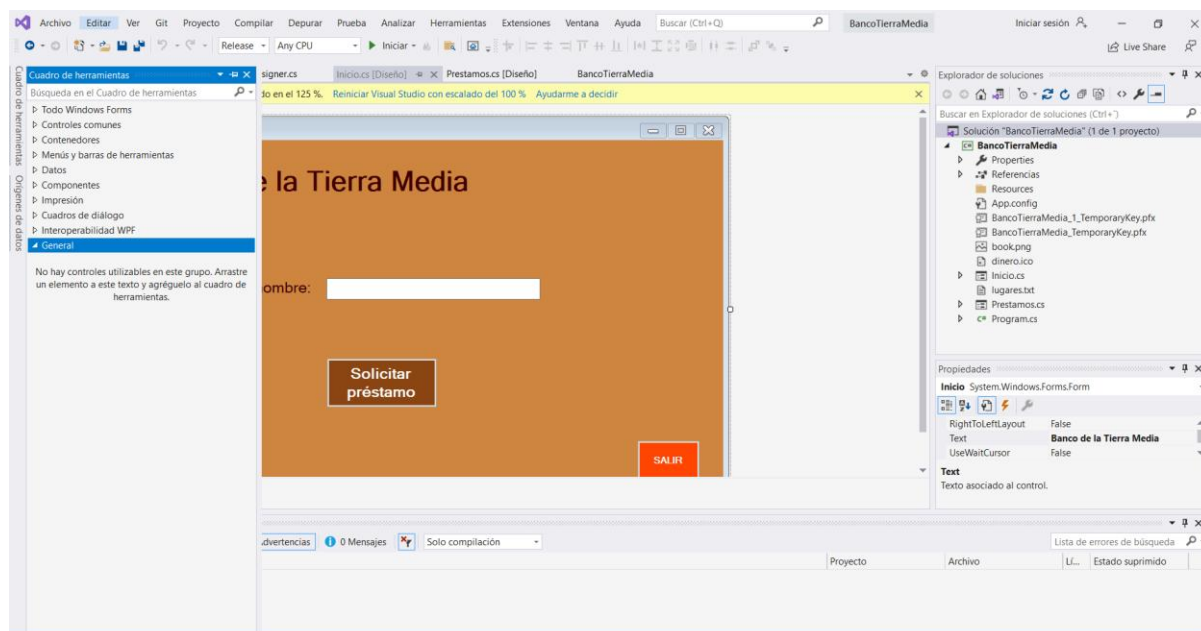
oficial.

Ejercicio:

Instalar Visual Studio siguiendo el anexo 2_Anexo_1-Instalacion Visual Studio.pdf
Si fuese necesario desinstalar se puede seguir el anexo 2_Anexo_2-Desinstalacion Visual Studio.pdf


Visual Studio tiene tres tipos de elementos que se reparten en su ventana principal:

- La parte superior de la ventana principal está compuesta por una barra de menús y barras de herramientas.
- La parte central de la ventana contiene la zona de trabajo en la que es posible visualizar los formularios o el código
- Varias ventanas secundarias representan las distintas herramientas a disposición del desarrollador. Estas ventanas se pueden colocar a la izquierda, derecha o debajo de la principal. Seleccionando el nombre de la ventana y arrastrando con el ratón nos aparecerá una guía de dónde se puede colocar. Cuando la ubicación de la ventana se corresponde con un anclajes se previsualiza el espacio que ocuparía mediante un fondo de color azul. Si se desvincula de la guía la ventana se convierte en flotante.



El entorno de trabajo puede gestionarse y adaptarse mediante el uso de tres opciones:

- El anclado de ventanas

	<p>UT 2 - <i>Generación de interfaces de usuario y Creación de componentes visuales</i></p>	<p>CFGS Desarrollo de Aplicaciones Multiplataforma</p> <p>Módulo: DDI</p>
---	---	---

- La ocultación automática de ventana de ventanas
- El uso de pestañas

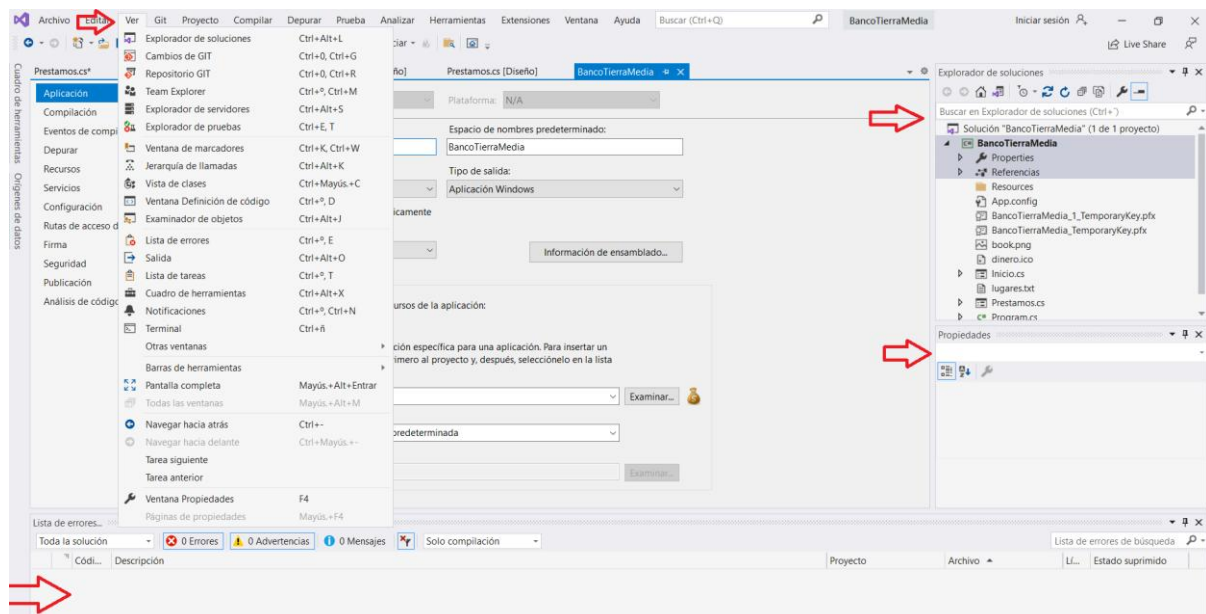
2.3 Soluciones y Proyectos en Visual Studio


Una solución es un contenedor para organizar uno o más proyectos de código relacionados.

Un proyecto es una respuesta a una necesidad planteada. Contiene todos los archivos que se compilan en un archivo ejecutable, biblioteca o sitio web. Estos archivos pueden incluir código fuente, iconos, imágenes, archivos de datos, etc. Un proyecto también contiene la configuración del compilador y otros archivos de configuración que podrían ser necesarios en diversos servicios o componentes con los que el programa se comunica. Al crear un proyecto, Visual Studio crea automáticamente una solución para este, a menos que ya haya una solución abierta.

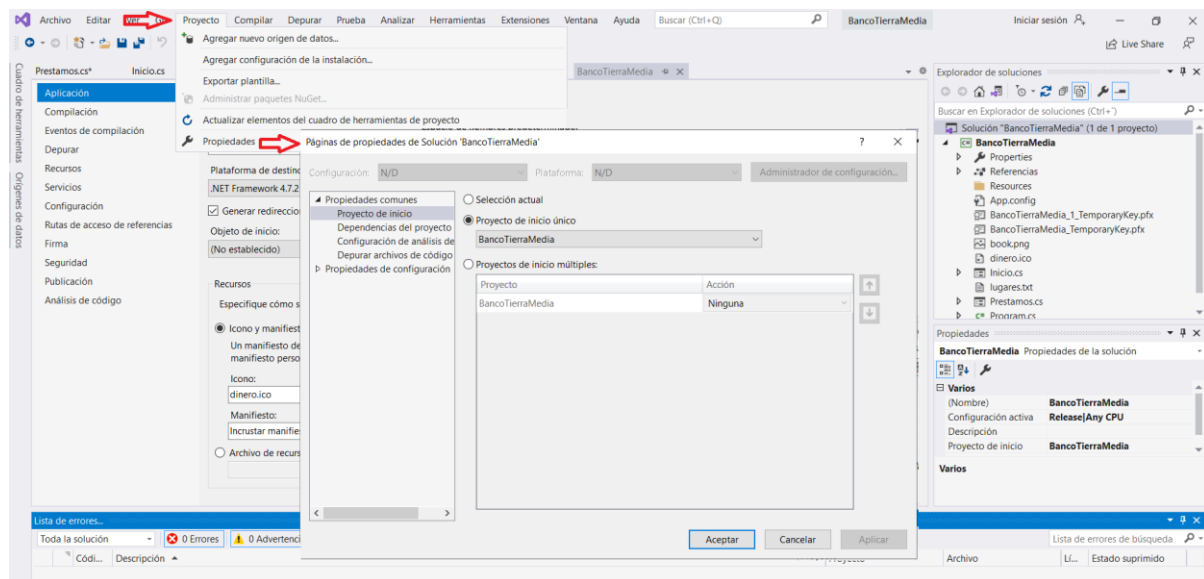
En Visual Studio la "carpeta de solución" es una carpeta virtual que solo se encuentra en el **Explorador de soluciones**.

En la interfaz de Visual Studio podemos incorporar diferentes ventanas de navegación que están disponibles en el menú "Ver", entre ellas el explorador de soluciones que nos facilitará la búsqueda de Soluciones y Proyectos y sus características y componentes:



	<p>UT 2 - Generación de interfaces de usuario y Creación de componentes visuales</p>	<p>CFGS Desarrollo de Aplicaciones Multiplataforma</p> <p>Módulo: DDI</p>
---	---	---

Podemos acceder a las propiedades de la solución pinchando sobre ella con el ratón y accediendo a Proyecto->Propiedades o bien pinchando sobre su nombre con el botón derecho del ratón y seleccionando igualmente propiedades. Este es un funcionamiento global de Visual Studio para los componentes de su interfaz, por lo que si queremos acceder a las propiedades de un proyecto debemos seguir los mismos pasos, pinchar sobre el nombre con el botón derecho del ratón o seleccionarlo (el elemento seleccionado aparecerá resaltado en pantalla) e ir a proyecto->propiedades.



Ejercicios:

Crea un proyecto siguiendo el 2_Anexo_3-Creacion Soluciones y Proyectos.pdf

Crea tu primera aplicación en consola

(<https://docs.microsoft.com/es-es/visualstudio/ide/quickstart-csharp-console?view=vs-2022>)

Podemos buscar un archivo de una solución en un equipo desde **Herramientas > Opciones > Proyectos y soluciones > Ubicaciones**.