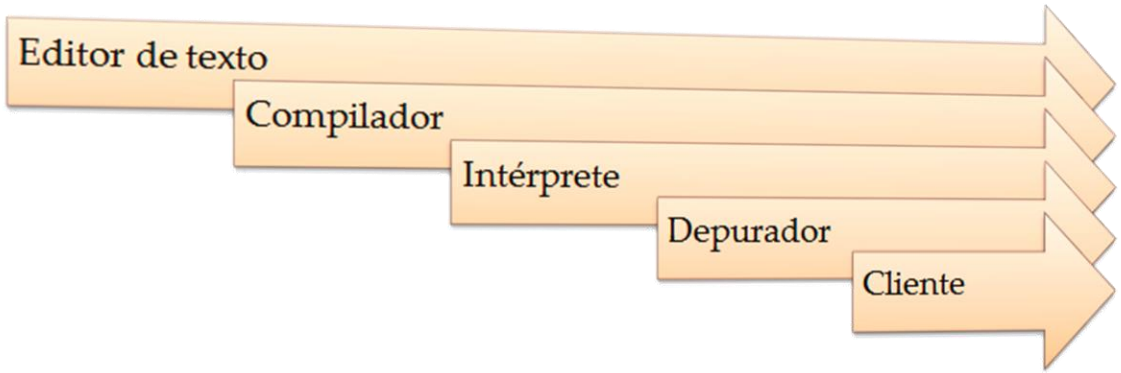
INSTALACIÓN Y USO DE ENTORNOS DE DESARROLLO

1. CARACTERÍSTICAS
2. CRITERIOS DE ELECCIÓN DE UN IDE
3. USO BÁSICO DE UN IDE
4. NUESTRA ELECCIÓN VISUAL STUDIO

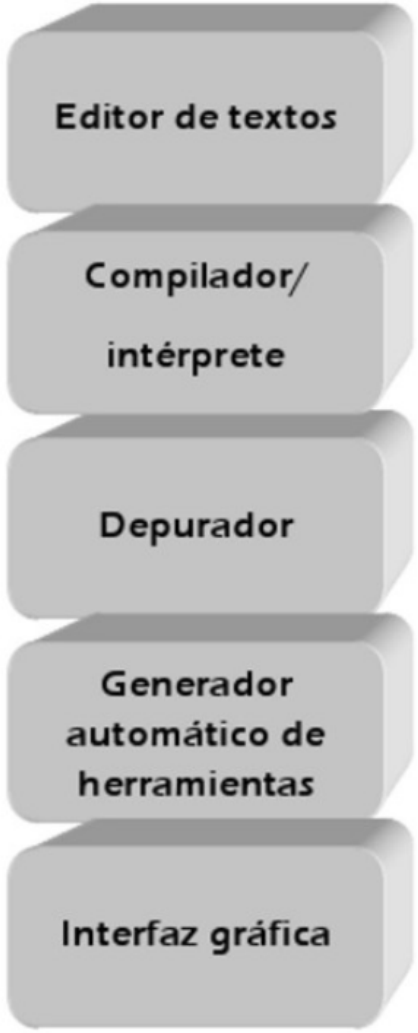
INSTALACIÓN Y USO DE ENTORNOS DE DESARROLLO

1. CARACTERÍSTICAS

Un entorno de desarrollo integrado o **IDE** (*Integrated Development Environment*) es un programa informático que tiene el objetivo de asistir al programador en la tarea de diseñar y codificar un software mediante la inclusión de múltiples herramientas destinadas para dicha tarea.



INSTALACIÓN Y USO DE ENTORNOS DE DESARROLLO



Editor de textos: Se resalta y colorea la sintaxis, tiene la función de autocompletar código, ayuda y listado de parámetros de funciones y métodos de clase. Inserción automática de paréntesis, corchetes, tabulaciones y espaciados.

Compilador/intérprete: Detección de errores de sintaxis en tiempo real. Características de refactorización.

Depurador: Botón de ejecución y traza, puntos de ruptura y seguimiento de variables. Opción de depurar en servidores remotos.

Generador automático de herramientas: Para la visualización, creación y manipulación de componentes visuales y todo un arsenal de asistentes y utilidades de gestión y generación código.

Interfaz gráfica: Nos brinda la oportunidad de programar en varios lenguajes con un mismo IDE. Es una interfaz agradable que puede acceder a innumerables bibliotecas y plugins, aumentando las opciones de nuestros programas.

INSTALACIÓN Y USO DE ENTORNOS DE DESARROLLO

1. CARACTERÍSTICAS

Además de las herramientas comunes que todo **IDE** debe tener (Editor de texto, compilador, depurador…), un **IDE** aporta una serie de herramientas adicionales para cumplir con mayor eficacia el objetivo de facilitar el trabajo a los desarrolladores.

Una de las herramientas más útiles e importantes es el autocompletado de código y las inspecciones de las clases y objetos. En **Visual Studio** esa herramienta se llama ***IntelliSense***.

Los ***snippets*** nos permiten utilizar código reusable de una manera rápida y cómoda. Algunos entornos de desarrollo admiten la posibilidad de crear tus propios ***snippets*** para utilizarlos cuando quieras.

INSTALACIÓN Y USO DE ENTORNOS DE DESARROLLO

1. CARACTERÍSTICAS

Como ya hemos comentado, la labor principal de un entorno de desarrollo es facilitar la vida al programador. Para cumplir éste propósito, los **IDEs** son altamente configurables y personalizables.

La configuración del IDE permite entre otras cosas añadir y modificar las barras de herramientas, pudiendo crear comandos personalizados y atajos de teclado para cada una de ellas. Estableciendo el posicionamiento de las ventanas y barras conjuntamente con los atajos de teclado podremos mejorar sumamente nuestro rendimiento y aprovechar con mayor comodidad todas las funciones del **IDE**.

La mayoría de los **IDEs** también permiten configurar interfaces diferentes dependiendo de la operación que se esté realizando, teniendo una configuración para la etapa de desarrollo y otra diferente para la etapa de depuración.

INSTALACIÓN Y USO DE ENTORNOS DE DESARROLLO

CRITERIOS DE ELECCIÓN DE UN IDE

- **SISTEMA OPERATIVO**

Sin lugar a dudas, uno de los criterios más restrictivos a la hora de seleccionar nuestro entorno de trabajo es saber en qué sistema operativo vamos a trabajar y, más importante aún, para qué sistema operativo vamos a desarrollar nuestro software.

Siempre y cuando nuestro software no vaya a ser ejecutado mediante una máquina virtual, estaremos desarrollando para un sistema operativo concreto.

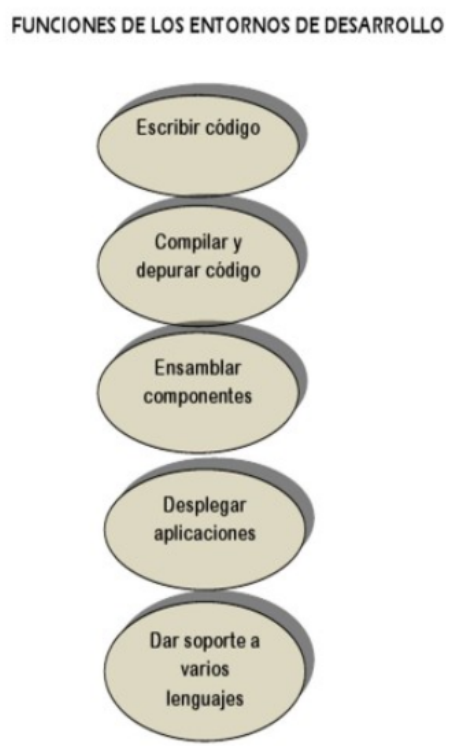
Esto realmente no se debe a una restricción inherente al **IDE**, sino al **compilador** que tiene el IDE integrado.

Este problema es fácilmente salvable compilando nuestro código fuente en un compilador de otro sistema operativo (siempre y cuando exista), por lo que dependiendo de nuestras necesidades pudiera ser un problema menor.

INSTALACIÓN Y USO DE ENTORNOS DE DESARROLLO

CRITERIOS DE ELECCIÓN DE UN IDE

Las funciones de los IDE son:



- ✓ Editor de código: coloración de la sintaxis.
- ✓ Auto-completado de código, atributos y métodos de clases.
- ✓ Identificación automática de código.
- ✓ Herramientas de concepción visual para crear y manipular componentes visuales.
- ✓ Asistentes y utilidades de gestión y generación de código.
- ✓ Archivos fuente en unas carpetas y compilados a otras.
- ✓ Compilación de proyectos complejos en un solo paso.
- ✓ Control de versiones: tener un único almacén de archivos compartido por todos los colaboradores de un proyecto. Ante un error, mecanismo de auto-recuperación a un estado anterior estable.
- ✓ Soporta cambios de varios usuarios de manera simultánea.
- ✓ Generador de documentación integrado.
- ✓ Detección de errores de sintaxis en tiempo real.

INSTALACIÓN Y USO DE ENTORNOS DE DESARROLLO

2. CRITERIOS DE ELECCIÓN DE UN IDE

No hay unos entornos de desarrollo más importantes que otros. La elección del IDE más adecuado dependerá del lenguaje de programación que vayamos a utilizar para la codificación de las aplicaciones y el tipo de licencia con la que queramos trabajar.

Tabla de los IDE más relevantes hoy en día:

Entorno de desarrollo	Lenguajes que soporta	Tipo de licencia
NetBeans	C/C++, Java, JavaScript, PHP, Python.	De uso público.
Eclipse	Ada, C/C++, Java, JavaScript, PHP.	De uso público.
Microsoft Visual Studio.	Basic, C/C++, C#.	Propietario.
C++ Builder.	C/C++.	Propietario.
JBuilder.	Java.	Propietario.

INSTALACIÓN Y USO DE ENTORNOS DE DESARROLLO

2. CRITERIOS DE ELECCIÓN DE UN IDE

Entornos Integrados Libres

Son aquellos con licencia de uso público.

No hay que pagar por ellos, y aunque los más conocidos y utilizados son Eclipse y NetBeans, etc..

IDE	Lenguajes que soporta	Sistema Operativo
NetBeans.	C/C++, Java, JavaScript, PHP, Python.	Windows, Linux, Mac OS X.
Eclipse.	Ada, C/C++, Java, JavaScript, PHP.	Windows, Linux, Mac OS X.
Gambas.	Basic.	Linux.
Anjuta.	C/C++, Python, Javascript.	Linux.
Geany.	C/C++, Java.	Windows, Linux, Mac OS X.
GNAT Studio.	Fortran.	Windows, Linux, Mac OS X.

INSTALACIÓN Y USO DE ENTORNOS DE DESARROLLO

2. CRITERIOS DE ELECCIÓN DE UN IDE

Entornos Integrados Propietarios

Son aquellos entornos integrados de desarrollo que necesitan licencia. No son free software, hay que pagar por ellos. El más conocido y utilizado es Microsoft Visual Studio, que usa el framework .NET y es desarrollado por Microsoft.

IDE	Lenguajes que soporta	Sistema Operativo
Microsoft Visual Studio.	Basic, C/C++, C#.	Windows.
FlashBuilder.	ActionScript.	Windows, Mac OS X.
C++ Builder.	C/C++.	Windows.
Turbo C++ profesional.	C/C++.	Windows.
JBuilder.	Java.	Windows.
JCreator.	Java.	Windows, Linux, Mac OS X.
Xcode.	C/C++, Java.	Mac OS X.

INSTALACIÓN Y USO DE ENTORNOS DE DESARROLLO

2. CRITERIOS DE ELECCIÓN DE UN IDE

- **LENGUAJE DE PROGRAMACIÓN Y FRAMEWORK**

Un IDE puede soportar uno o varios lenguajes de programación, por lo que saber en qué lenguaje de programación vamos a codificar nuestro software y qué lenguajes nos ofrecen los distintos **IDE** es una información valiosa que hay que tener en cuenta.

Lo mismo ocurre con las plataformas de trabajo, también llamadas ***framework***.

Este criterio va de la mano con el sistema operativo, ya que si quisiéramos desarrollar en **Visual Basic** bajo un sistema operativo **Linux** no sería **Visual Studio** nuestra opción, sino que tendríamos que utilizar **Gambas**.

INSTALACIÓN Y USO DE ENTORNOS DE DESARROLLO

2. CRITERIOS DE ELECCIÓN DE UN IDE

- **HERRAMIENTAS Y DISPONIBILIDAD**

Las diferentes herramientas de las que disponen los IDE son el último criterio de selección, seguramente nos encontremos con varios IDE que cumplen los requisitos de lenguaje y sistema operativo, pero no todos tienen las mismas funciones, por lo que saber cuáles son esas herramientas es un dato sumamente importante en nuestra decisión.

En ocasiones pueden ser restrictivos ya no solo por tus propias preferencias, sino por trabajar de manera colaborativa y el modo de utilizar e interpretar diferentes códigos entre diferentes IDE.

Los entornos de desarrollos generan código automáticamente siguiendo un patrón propio que suele ser único. Es por ello que realizar un proyecto de forma colaborativa donde se utilice el mismo lenguaje y el mismo framework puede resultar una experiencia diferente dependiendo del entorno de desarrollo que se utilice.

INSTALACIÓN Y USO DE ENTORNOS DE DESARROLLO

2. CRITERIOS DE ELECCIÓN DE UN IDE

- **HERRAMIENTAS Y DISPONIBILIDAD**

Fuera del marco de trabajo colaborativo, también tendremos nuestras preferencias y necesidades, por lo que, si necesitamos tener una funcionalidad concreta, deberemos encontrar un IDE que nos la facilite.

Podríamos también ir más allá de las meras funcionalidades añadidas e irnos a un ámbito mucho más centrado en el propio software, como podría ser la interfaz de usuario, los IDE pueden incluir sus propios y específicos controles que mejoran la interfaz y aportan mayor funcionalidad y usabilidad a nuestros formularios y aplicaciones. También podríamos ver este criterio desde un punto de vista más destinado a la codificación y pensar en qué refactorizaciones automáticas nos podemos encontrar entre los IDE a la hora de elegirlo.

Una vez comprobados todos los criterios de selección mencionados tendríamos que comprobar si el IDE que cumple los requisitos está a nuestro alcance, ya sea por una cuestión de presupuesto o localización.

INSTALACIÓN Y USO DE ENTORNOS DE DESARROLLO

3. USO BÁSICO DE UN IDE

Parece obvio pensar que el uso básico de un IDE consiste en desarrollar software, y no sería un pensamiento equivocado. No obstante, esa misma la tarea la podríamos realizar con un editor de texto y un compilador, por lo que el uso de un IDE va más allá de la simple edición de código.

Muchas de las herramientas más habituales que solemos usar conjuntamente con los IDE también se encuentran disponibles fuera de ellos, como las herramientas de modelado o de pruebas unitarias. En esencia, todas esas aplicaciones pueden o podrían estar disponibles sin necesidad de un entorno de desarrollo, o mejor dicho, sin que fuese un entorno de desarrollo integrado pero nos veríamos forzados en ese caso a utilizar diferentes aplicaciones de manera simultánea, que además, no tendrían por que comunicarse de manera eficiente entre ellas.

Es por ello, que la verdadera usabilidad del entorno de desarrollo consiste una vez más en ofrecer una comodidad y una asistencia a un trabajo que podríamos hacer sin él, del mismo modo que podríamos subir al noveno piso de un edificio sin ascensor.

INSTALACIÓN Y USO DE ENTORNOS DE DESARROLLO

3. USO BÁSICO DE UN IDE

La necesidad básica que todo IDE debe cubrir es la creación o edición de programas y convertir ese código fuente en código ejecutable.

Un IDE realiza esa operación de manera conjunta y compacta. Gracias a un gestor de proyectos podemos ajustar las dependencias de cada sección de nuestro programa y todas las necesidades y opciones de compilación que queramos.

Los IDE, además, suelen ofrecer una funcionalidad añadida, ya que permiten ejecutar de manera virtual el programa que se está codificando en cualquier momento siempre y cuando no tenga errores de compilación.

INSTALACIÓN Y USO DE ENTORNOS DE DESARROLLO

3. USO BÁSICO DE UN IDE

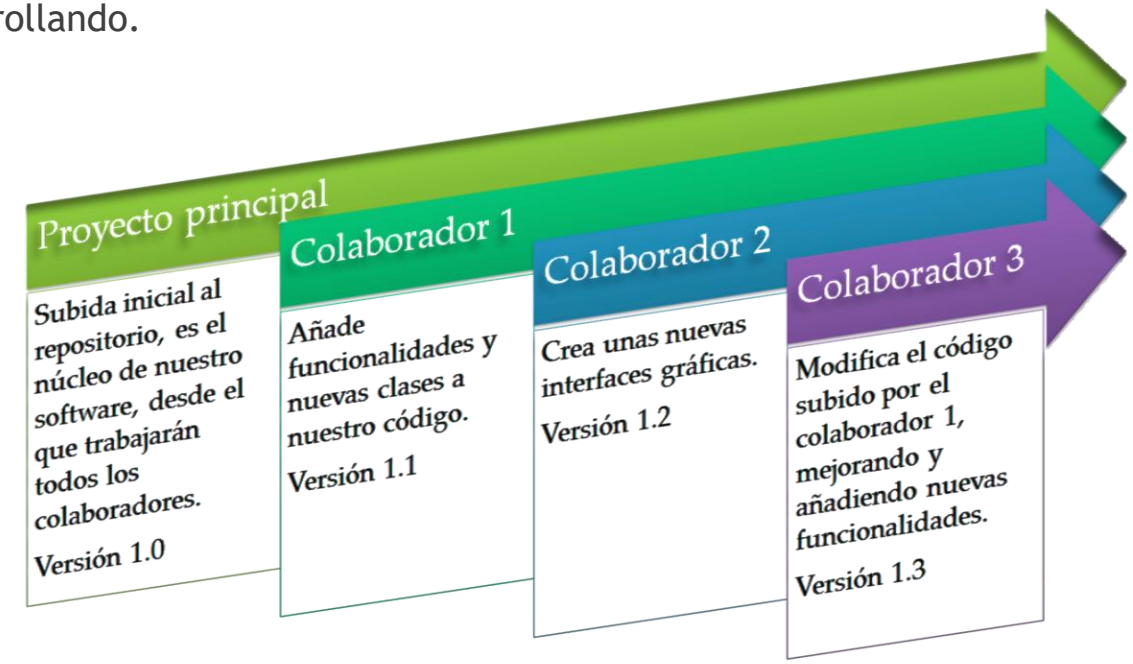
Los proyectos sobre los que vamos a trabajar en un IDE pueden requerir de un grupo de trabajo, y ese grupo de trabajo puede requerir un proyecto de **desarrollo colaborativo**.

Para ello haremos uso de programas de **control de versiones** integradas en el entorno de desarrollo. Los programas de control de versiones son aplicaciones que constan de servidor y cliente, donde en la parte del servidor se crean repositorios para que los clientes puedan descargar y subir código. Son herramientas asíncronas que permiten controlar y gestionar las fuentes y versiones del código del repositorio.

Gracias a las herramientas del **IDE**, podemos hacer un uso mucho más rápido y avanzado de los controles de versiones, pudiendo elegir qué archivos actualizar en cualquier lado de nuestra conexión (servidor o cliente), omitir cambios para no pisar nuestro trabajo con el de otros, y viceversa, y una gran cantidad de operaciones de la misma índole.

3. USO BÁSICO DE UN IDE

La sincronización de un IDE con el repositorio del proyecto permite además saber qué archivos han cambiado y por ende tener un control de versiones más allá del servidor, tenerlo directamente sobre el código que estamos actualmente desarrollando.



INSTALACIÓN Y USO DE ENTORNOS DE DESARROLLO

4. NUESTRA ELECCIÓN VISUAL STUDIO

Hemos visto que existe una gran complejidad en la elección de un entorno de desarrollo debido a que los criterios de selección se complementan y vienen relacionados entre sí de forma muy habitual.

Siguiendo los criterios que hemos aprendido en este capítulo, hemos elegido utilizar el **Visual Studio Ultimate 2010** como entorno de desarrollo. Visual Studio es uno de los entornos de desarrollo más pulidos, profesionales y completos que podemos encontrar en el mercado, además, tiene a sus espaldas una excelente plataforma de trabajo: el *framework* **.NET**.

4. NUESTRA ELECCIÓN VISUAL STUDIO

RECORRIDO POR LAS VENTANAS Y PALETAS PRINCIPALES

Página principal personalizada: Disponemos en primera plana de una página principal con información de actividad reciente que se puede personalizar para mostrar últimas noticias, tutoriales y guías, o simplemente información adicional sobre el IDE.

Explorador de soluciones: Nos permite además ver las referencias, conexiones de datos y dependencias de los diferentes proyectos, pudiendo establecer propiedades adicionales a las mismas.

Editor de diseño: Con el editor de diseño, podremos crear y posicionar los controles que nuestra interfaz necesita.

Editor de código: El editor de texto del IDE donde codificaremos nuestro software.

4. NUESTRA ELECCIÓN VISUAL STUDIO

RECORRIDO POR LAS VENTANAS Y PALETAS PRINCIPALES

Editor de vista compartida: La vista compartida nos permite partir la pantalla de forma horizontal o vertical y modificar la posición de la línea divisoria para que se ajuste a nuestras necesidades.

Consola de compilación: Disponemos de una consola de compilación donde revisar los procesos y errores ocurridos durante dicho proceso.

Ventanas de depuración: Visual Studio nos ofrece una serie de pantallas específicas, visibles durante la depuración del programa con el objetivo de ofrecer toda la información necesaria de la manera más compacta mientras dura la ejecución de depuración.

4. NUESTRA ELECCIÓN VISUAL STUDIO

PERSONALIZACIÓN Y CONFIGURACIÓN

Ventanas: Podemos añadir, quitar, acoplar y mover las ventanas de la forma que queramos de una manera muy sencilla. Si queremos añadir una ventana al entorno de desarrollo, no tenemos más que ir al menú **Ver > Otras Ventanas** y escoger la que queramos.

Ventanas de documentos: Se pueden colocar como una ventana independiente, como una parte de la organización de fichas (como si fuesen pestañas) o como parte de un grupo dividido de ventanas o grupo dividido de fichas.

Ventanas de herramientas: Cualquier ventana de herramientas puede posicionarse en cualquier posición posible, incluyendo las pestañas o subbloques de los bloques de ventanas.

Barras de herramientas: Tenemos diversas barras de herramientas a nuestra disposición en los entornos de desarrollo. Se pueden añadir, quitar, mover y modificar. Para personalizar o crear nuevas barras de herramientas, utilizaremos la opción del menú **Herramientas > Personalizar**, y dentro de la ventana **Personalizar**, en **Comandos.**

4. NUESTRA ELECCIÓN VISUAL STUDIO

- **PERSONALIZACIÓN Y CONFIGURACIÓN**

Opciones del entorno: Disponemos de un menú de opciones de personalización en **Herramientas > Opciones** en el que una vez activada la opción de **Mostrar todas las configuraciones** tendremos acceso a todas las opciones básicas que nos ofrece el entorno de desarrollo.

Opciones del proyecto: Dependiendo del tipo de proyecto, también nos podremos encontrar con diferentes opciones dentro de las propiedades del proyecto.

Para acceder a las propiedades del proyecto, deberemos hacer clic con el botón derecho en el proyecto y seleccionar **Propiedades** en el menú contextual emergente que nos aparece.