Unidad 2

Instalación y uso de entornos de desarrollo

Entornos de desarrollo

Índice

Introducción

Para crear aplicaciones, los programadores utilizan los entornos de desarrollo. Realmente, utilizando simplemente un editor y un compilador, podemos desarrollar este tipo de programas, pero en el mundo profesional, se usa un IDE casi siempre.

Las herramientas de un IDE son la siguientes:

- **Editor.** Normalmente, estos editores colorean el texto en función de la sintaxis del lenguaje utilizado, así el programador puede detectar fallos en el código con mayor facilidad.
- **Compilador o intérprete.** En función del lenguaje que utilicemos, tendremos que hacer uso de un compilador para la generación de código ejecutable.
- **Depurador (intérprete).** Los depuradores de calidad cuentan normalmente con un intérprete detrás que irá ejecutando paso a paso las distintas órdenes, controlará el valor de las variables, etc.
- **Constructor de interfaces gráficos.** Es utilizado por el desarrollador para la creación de tablas, literales, botones, pestañas, etc. Podemos encontrar en ellos todos los componentes propios de una interfaz.

Al finalizar esta unidad

- Aprenderemos como instalar un IDE y como configurarlo para sacarle todo el rendimiento que este ofrece.
- Conoceremos los IDE más exitosos de los últimos tiempos.
- Sabremos depurar cualquiera código.
- Podremos genera documentación de forma automática
- Conoceremos como ampliar las capacidades del IDE mediante plugins.

2.1. Los primeros entornos de desarrollo

2.1.1. Turbo Pascal

Turbo Pascal fue creado en el año 1983 por la empresa Borland y en su momento fue el IDE más potente. En sus inicios trabajaba en CP/M y CP/M 86, MS-DOS y Macintosh, pero más tarde, lanzaron una versión para Windows que triunfó bastante.

Siete fueron las versiones que salieron, y el ratón podía utilizarse en las últimas. En un mismo editor podías abrir diferentes archivos (diferentes ventanas) y los podías programar orientado a objetos. Además, contaba con Turbo Profile, una herramienta de optimización de código.

Constituyó una revolución para su tiempo, ya que la velocidad a la que compila es incluso superior a los actuales compiladores.

El éxito que obtuvo esta herramienta hizo que Borland se lanzara a la creación de otras herramientas (Delphi) que se basaban en el mismo lenguaje: Pascal.

2.1.2. Visual Basic 6

Uno de los IDE más utilizados en su momento, posiblemente el que más, fue Visual Basic 6. Junto con esta herramienta surgió un nuevo paradigma de desarrollo conocido como RAD (rapid application development), en español, desarrollo rápido de aplicaciones. En este paradigma, en primer lugar, se hacía rápidamente una interfaz, y posteriormente se testeaba con el usuario. Una vez que el usuario aprobaba dicha interfaz, se procedía a la creación de la base de datos y del código. Esto modificó la forma en que se programaba.

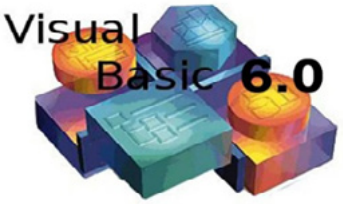

Imagen 1. Logo de Visual basic 6.

Partiendo de ciertos componentes ofrecidos por la herramienta, los programadores desarrollaban interfaces. Incluso permitía el uso de componentes externos, por lo que facilitaba el desarrollo.

Para acceder a las bases de datos se utilizaba ActiveX Data Objects, RDO o DAO, siendo el primero de todos el mejor.

Actualmente, Visual Basic se sigue utilizando ya que las macros creadas con Office usan un dialecto suyo, el Visaul Basic for applications (VBA). Las macros constituyen una herramienta de lo más útil y potente, ya que cuentan con propiedades de Office a la par que con el poder de desarrollo que tienen los lenguajes orientados a objetos.

2.1.3. Delphi

Es verdad que Turbo Pascal fue un grande en su momento, pero otro de los gigantes de la informática, Microsoft, creó Visual Basic, un IDE para Windows que consiguió que Borland retrasara su lanzamiento de Delphi, el cual fue como la continuación de Turbo Pascal, pero adaptándolo al sistema Windows, de la misma manera que Builder C++ fue la continuación de Turbo C.

A parte de Delphi, Borland creó JBuilder, que se trataba de otro IDE, pero esta vez de Java, cuya ventaja era que estaba también para Linux.

El hermano que tuvo Delphi para Linux se llamó Kylix, pero este solo se mantuvo hasta la versión 3.0. La ventaja de esta herramienta era que podías recompilar en Linux cualquier proyecto desarrollado en Windows y al revés, mientras que hubieran utilizado controles estándar).

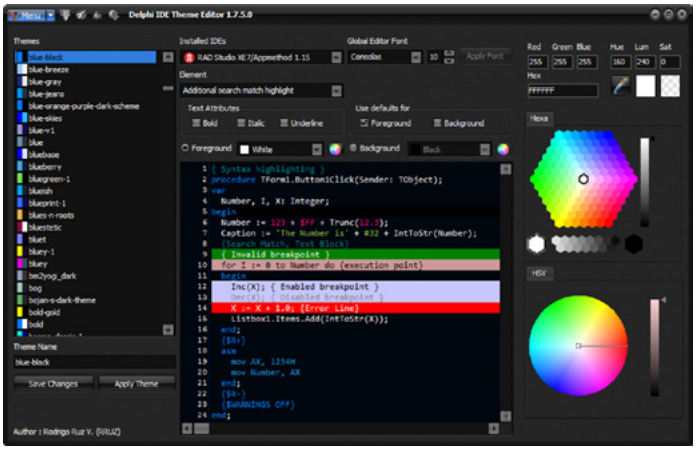

Imagen 2. Delphi IDE. Fuente: artificialroutine.com

2.1.4. Visual C++

Con este IDE podemos programar tanto en C++ como en C. Su punto fuerte es que contiene las bibliotecas propias de Windows, el framework .NET y las Microsoft Foundation classes (MFC).

Este IDE es muy potente ya que podemos incluso añadir otras bibliotecas externas como SDL, DirectX y wxWidgets.

Y, además, .NET cuenta con el recolector de basura o garbage collector como Java.

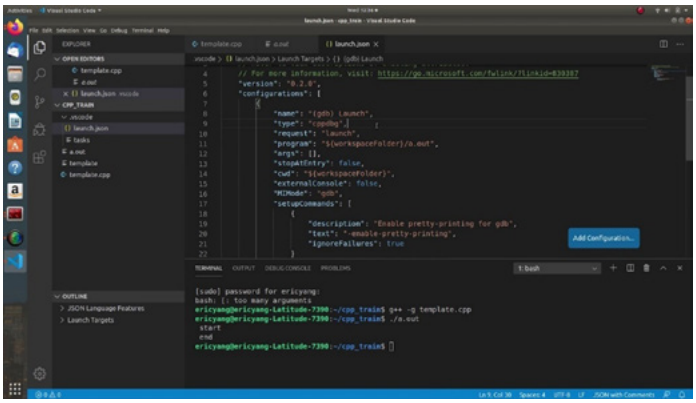

Imagen 3. Visual C++ IDE.

2.2. Entornos de desarrollo actuales

2.2.1. Xcode

Con Xcode podemos desarrollar aplicaciones destinadas a dispositivos Apple. Esta herramienta nos permite llevar a cabo aplicaciones nativas de OS X y iOS.

Siendo desarrollador de Apple es posible obtener las versiones antes de que sean lanzadas al mercado. En la actualidad, hacerse desarrollador de Apple es gratuito, únicamente hay que pagar una subscripción anual en caso de querer subir aplicaciones a la App Store.

Las últimas versiones nos dejan programar con Swift, pero las versiones más antiguas, solo dejan programar con Objective C. Este último es un lenguaje similar a Java/C/C++, pero la sintaxis difiere ligeramente. Este lenguaje es bastante potente y también está orientado a objetos.

Imagen 4. Logo de Xcode.

2.2.2. NetBeans

NetBeans está disponible para una gran variedad de sistemas operativos (Linux, Windows o Mac OS X), esto es debido a que está escrita en Java. Pero a parte de Java, también podemos desarrollar con otros lenguajes como C/C++, HTML5, PHP y Python.

Una gran cantidad de programadores se decantan por NetBeans ya que es open source.

Está basado en la modularidad, es decir, el programador puede añadir módulos que realizan una serie de operaciones en función de lo que se necesite. Es más, al descargarlo, este viene con todos los módulos propios de Java. Sun Java Studio Creator y Sun Studio son algunas de las herramientas que se basan en NetBeans.

Para desarrollar interfaces de usuario, cuenta con una herramienta que nos posibilita la creación de aplicaciones mediante la librería Swing. Esta aplicación al principio se conocía como Matisse.

Y, por último, podemos programar en Ajax, CSS y JavaScript utilizando el editor.

Imagen 5. Logo de NetBeans.

2.2.3. Eclipse

Este IDE es de open source. Presenta una plataforma potente, con un compilador, depurador y editor respetables (el ECJ). El Java development toolkit (JDT), es uno de los más potentes del mercado y, además, cuenta con una gran comunidad que aportan continuamente mejoras al software.

Este evolucionó de VisualAge, desarrollado por IBM, pero hoy en día, Eclipse, sin ánimo de lucro y de forma independiente, es quien mantiene su desarrollo.

Antes contaba con una licencia del tipo CPL (common public license), pero hoy en día esta licencia es EPL (Eclipse public license).

Imagen 6. Logo de Eclipse IDE.

2.3. Entornos de desarrollo online

Este tipo de entornos cada vez se utilizan más, esto es debido a que, a pesar de la menor potencia, permiten trabajar de manera colaborativa, trabajar desde cualquier dispositivo, el uso de repositorios comunes, etc.

Es por todo esto que empresas y desarrolladores eligen esta opción.

2.4. Entornos de desarrollo libres y propietarios

Tenemos a nuestra disposición numerosos IDEY, y estos cuentan con más o menos opciones dependiendo de la cantidad de usuarios que tengan:

	IDE	Licencia	Windows	Linux	Mac OS X
Lenguaje Java	Eclipse	EPL	Sí	Sí	Sí
	NetBeans	CDDL/GPL2	Sí	Sí	Sí
	Visual Studio	Propietario	Sí	No	No
	JDeveloper	Propietario	Sí	Sí	Sí
Lenguaje JavaScript	Eclipse	EPL	Sí	Sí	Sí
	NetBeans	CDDL/GPL2	Sí	Sí	Sí
	Geany	GPL	Sí	Sí	Sí
	KDevelop	GPL	No	Sí	No
	JBuilder	Propietario	Sí	Sí	Sí
	JCreator	Propietario	Sí	No	No
	JDeveloper	Propietario	Sí	Sí	Sí

Imagen 7. IDE para los lenguajes Java y JavaScript.

2.5. Instalación de un entorno integrado de desarrollo

2.5.1. El compilador de Java

Dentro del JDK (Java development kit) o equipo de desarrollo Java, encontramos el compilador de Java, conocido como javac. Este compilador es totalmente necesario para programar en Java, por lo que necesitaremos que en nuestra máquina esté instalado un JDK.

El sistema deberá además contar con un JRE (Java runtime environment) o entorno de ejecución Java, en el que estará incluido un JVM (Java virtual machine) o máquina virtual Java.

Gracias a que Java es multiplataforma, al compilar el programa en un sistema, este funcionará en los demás sistemas mientras que estos cuenten con la JVM correcta. Por ello, debemos de ser conscientes de que cada sistema tiene su propia JVM.

2.5.2. Dudas frecuentes sobre el compilador de Java

Cómo sé si ya está instalado el JVM

Podremos saberlo ejecutando este comando:

```
$ java -version
java version "1.8.0_91"
Java(TM) SE Runtime Environment (build 1.8.0_91-b15)
Java HotSpot (TM) Client VM (build 25.91-b15, mixed mode)
```

Cómo sé si ya está instalado el JDK

Podremos saberlo con este comando:

```
$ javac -version
javac 1.8.0_05
```

Qué hay que hacer para instalar el JRE y el JDK

Ubuntu cuenta con una versión de JDK y JRE, para instalarla hay que introducir el siguiente comando:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install default-jre
$ sudo apt-get install dejault-jdk
```

Primero introducimos dos comandos para que, en caso de no estar actualizados los paquetes y el sistema, estos se actualicen.

Y después los dos últimos comandos son para instalar el JRE y el JDK respectivamente.

2.6. Depurar un programa

Los programas suelen tener fallos y evolucionan con el tiempo, por eso es necesario depurarlos para asegurar su correcto funcionamiento. A continuación, vamos a ver los pasos a seguir a la hora de depurar un programa hecho con NetBeans.

Paso 1

Abrimos NetBeans y ejecutamos el proyecto que vamos a depurar. En caso de no contar con un programa a depurar, podemos ir a Archivo y nuevo proyecto, y seleccionamos la opción de Aplicación Java. Hayas creado un nuevo proyecto o hayas abierto uno propio, podremos ver el código fuente en la ventana principal del editor.

Paso 2

Se crea un punto de interrupción de alguna línea del código de manera aleatoria. Este punto de interrupción se trata de un punto en concreto del programa donde se detendrá la ejecución del código y se detiene hasta que el depurador realice su función. Esta técnica resulta bastante eficaz a la hora de tener una idea más clara de que procesos realiza el programa en cada momento. Si creemos que el código puede generar errores, entonces lo correcto es poner el punto de interrupción donde creamos que hemos fallado. Si queremos establecer un punto de interrupción haremos click en un número de línea del lado izquierdo del código del programa. Una vez se realiza esto nos aparecerá un menú donde habrá que seleccionar "Punto de interrupción/alternar puntos de interrupción de líneas". Así ya habríamos creado este punto de interrupción.

Paso 3

Debemos de teclear "Ctrl + F5" y comenzará el proyecto de depuración. El depurador ejecutará el programa y se parará en el punto de ejecución. Ahora colocamos el ratón sobre las variables y aparecerán junto a ellas unas ventanas de información que nos dirán de que tipo es la variable y su contenido. En la ventana superior derecha de NetBeans podemos ver que se muestra cuanta memoria es usada por el programa de manera actual.

Paso 4

Saltamos a la línea siguiente de nuestro código pulsando F7 o F8, cualquiera vale. Si pulsamos F7 el depurador se mete en el código y si pulsamos F8, pasamos a través del código.

Nos referimos a meternos en el código a mostrar cómo funcionan las llamadas a funciones y se profundiza en esto si nos encontramos con multitud de llamadas sucesivas. En cambio, pasar a través del código quiere decir que no hace caso a las llamadas a funciones y solo se fija en el valor que se devuelve. Y con esto, habríamos terminado con la depuración de un programa.

2.7. Profiler. Análisis de aplicaciones

Hay ocasiones en las que cuando una aplicación está finalmente desarrollada o todavía en periodo de pruebas, se tiene que ver como rinde en un entorno de producción. NetBeans nos ayuda a realizar esta prueba ya que trae una herramienta llamada Profiler que se encarga de monitorizar el rendimiento de la CPU, etc…

De momento no vamos a trabajar con ella, por lo que solo es necesario conocerla.

2.8. Generación automática de documentación

Para que se pueda mantener el código de manera correcta y poder implementar futuras necesidades, es necesario que haya una documentación de cada programa o aplicación.

Un ejemplo es Java, donde la documentación se añade en el mismo código gracias a su lenguaje. Además, existe una herramienta que recibe el nombre de Javadoc que recoge los comentarios recogidos en el código y los transforma a HTML para su visionado en WEB.

En este mismo módulo, pero más adelante, veremos esto en más profundidad.

2.9. Gestión de módulos

Los entornos de desarrollo aumentan su capacidad de trabajo gracias a que tienen unos ciertos módulos o plugins que ayudan a la hora de trabajar con otros lenguajes de programación etc.

2.9.1. Añadir un módulo nuevo

Vamos a ver ahora como añadir módulo nuevo paso a paso. Para llegar a este propósito podemos o bien realizar todas las conexiones manualmente o, loque vamos a hacer ya que es más sencillo, usar el esqueleto que es generado por Netbeans Platform.

1. En el menú de Netbeans, seleccionamos: **New → New Project**.
2. Dentro del árbol de proyectos seleccionaremos la categoría **Netbeans Modules** que se encuentra en la carpeta **Java with Ant**.
3. Seleccionamos la opción **Module**.

   Module Suite incluirá los módulos añadidos junto con otros extras que van de manera nativa en el Entorno de desarrollo. Se podrán añadir más adelante más funciones sin problemas.

4. Ahora pulsamos **Next** y se pasa al siguiente paso del asistente de adición de módulos en Netbeans.

   En la izquierda se nos muestra un listado del proceso y en la parte inferior contamos con algunos elementos que nos ayudan a interactuar directamente con el asistente.

5. Ahora rellenamos el nombre y dejamos los valores predefinidos.

6. Podemos comprobar que cuando se haya creado el módulo, se añadirá de manera automática al listado de módulos que aparecen en **Netbeans Platforma Application**.

7. En un primer campo contamos con tener que añadir la jerarquía del código.

8. Una vez que termina el asistente, el módulo también se añade al listado de los proyectos abiertos y se vinculará a la aplicación principal.

2.9.2. Eliminar un módulo

Si creemos que tenemos un módulo o plugin que ya no es necesario, tenerlo solo hará que haya un espacio ocupado en el disco reservado para el entorno. Cuando tenemos esta situación, muchas veces sería conveniente eliminarlo.

Para eliminar un módulo en Netbenas tendremos que realizar los pasos siguientes:

1. Buscar el módulo dentro de la lista de los módulos instalados en el IDE.
2. Eliminarlo, y para ello hay dos opciones:
   a. Desactivarlo: seguirá instalado, pero no se puede usar, de hecho, no aparecerá.
   b. Desinstalarlo: se elimina el módulo de manera permanente y si queremos usarlo deberíamos de volver a instalarlo, desaparece de manera física de memoria.

Desde la ventana siguiente podemos ver que el gestor de complementos que instala NetBeans en Ubuntu nos avisa cuando queramos eliminar un módulo:

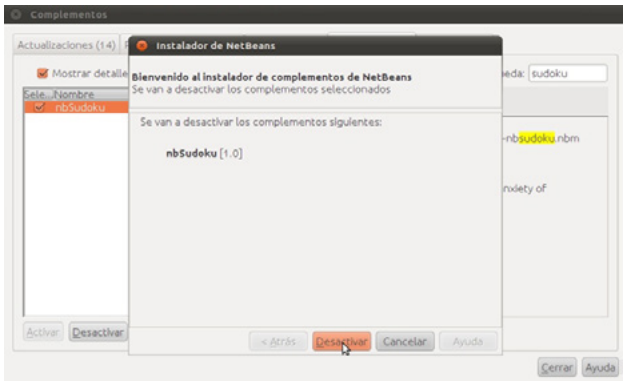

Imagen 8. Ventana de aviso de NetBeans.

Siempre vamos a tener la opción de desactivar o desinstalar y en el ejemplo, cómo podemos ver, se opta por su desactivación frente a la desinstalación.

www.universae.com