

**EN**

**02**

**Instalación y uso de entornos de  
desarrollo.**

---

# 1.- Concepto de entorno de desarrollo. Evolución histórica.

---

En la unidad anterior tratamos las fases en el proceso de desarrollo de software.

Una de ellas era la fase de **codificación**, en la cual se hacía uso de algún lenguaje de programación para pasar todas las acciones que debía llevar a cabo la aplicación a algún lenguaje que la máquina fuera capaz de entender y ejecutar.

También se hizo alusión a herramientas de apoyo al proceso de programación.

En esta unidad vamos a analizar, instalar y ejecutar estas herramientas para entender su acción y efecto.

Muchas personas aprenden a programar utilizando un [editor de texto simple](#), [compilador](#) y [depurador](#). Pero la mayoría, finalmente, terminan haciendo uso de algún entorno de desarrollo integrado para crear aplicaciones.

Un entorno integrado de desarrollo ([IDE](#)), es un tipo de software compuesto por un conjunto de herramientas de programación. En concreto, el IDE se compone de:

- ✓ Editor de código de programación.
- ✓ Compilador.
- ✓ Intérprete.
- ✓ Depurador.
- ✓ Constructor de interfaz gráfico.

Los primeros entornos de desarrollo integrados nacieron a principios de los años 70, y se popularizaron en la década de los 90. Tienen el objetivo de ganar fiabilidad y tiempo en los proyectos de software. Proporcionan al programador una serie de componentes con la misma interfaz gráfica, con la consiguiente comodidad, aumento de eficiencia y reducción de tiempo de codificación.

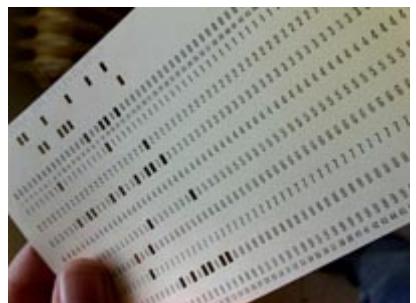
Normalmente, un IDE está dedicado a un determinado lenguaje de programación. No obstante, las últimas versiones de los IDE tienden a ser compatibles con varios lenguajes (por ejemplo, Eclipse, NetBeans, Microsoft Visual Studio) mediante la instalación de plugins adicionales.

En este tema, nuestro interés se centra en conocer los entornos de desarrollo, los tipos, en función de su licencia y del lenguaje de programación hacia el cual están enfocados. Instalaremos NetBeans bajo Ubuntu y veremos cómo se configura y cómo se generan ejecutables, haciendo uso de sus componentes y herramientas.

## 1.1.- Evolución Histórica.

---

En las décadas de utilización de la [tarjeta perforada](#) como sistema de almacenamiento el concepto de **Entorno de Desarrollo Integrado** sencillamente no tenía sentido.



Los programas estaban escritos con [diagramas de flujo](#) y entraban al sistema a través de las **tarjetas perforadas**. Posteriormente, eran compilados.

El primer lenguaje de programación que utilizó un IDE fue el [BASIC](#) (que fue el primero en abandonar también las tarjetas perforadas o las cintas de papel).

Este primer IDE estaba basado en consola de comandos exclusivamente (normal por otro lado, si tenemos en cuenta que hasta la década de los 90 no entran en el mercado los sistemas operativos con interfaz gráfica). Sin embargo, el uso que hace de la gestión de archivos, compilación y depuración; es perfectamente compatible con los IDE actuales.

A nivel popular, el primer IDE puede considerarse que fue el IDE llamado **Maestro**. Nació a principios de los 70 y fue instalado por unos 22.000 programadores en todo el mundo. Lideró este campo durante los años 70 y 80.

El uso de los entornos integrados de desarrollo se ratifica y afianza en los 90 y hoy en día contamos con infinidad de IDE, tanto de licencia libre como no.

## Tipos de entornos de desarrollo más relevantes en la actualidad.

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.

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.

## 2.- Funciones de un entorno de desarrollo.

Como sabemos, los entornos de desarrollo están compuestos por una serie de herramientas software de programación, necesarias para la consecución de sus objetivos. Estas herramientas son:

- ✓ Un editor de código fuente.
- ✓ Un compilador y/o un intérprete.
- ✓ Automatización de generación de herramientas.
- ✓ Un depurador.

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.



Otras funciones importantes son:

- ✓ Ofrece refactorización de código: cambios menores en el código que facilitan su legibilidad sin alterar su funcionalidad (por ejemplo cambiar el nombre a una variable).
- ✓ Permite introducir automáticamente tabulaciones y espaciados para aumentar la legibilidad.
- ✓ Depuración: seguimiento de variables, puntos de ruptura y mensajes de error del intérprete.
- ✓ Aumento de funcionalidades a través de la gestión de sus módulos y plugins.
- ✓ Administración de las interfaces de usuario (menús y barras de herramientas).
- ✓ Administración de las configuraciones del usuario.

### 3.- Entornos integrados libres y propietarios.

#### 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, hay bastantes más.

#### Tipos de entornos de desarrollo libres más relevantes en la actualidad.

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.

El aspecto de la licencia del IDE que se elija para el desarrollo de un proyecto es una cuestión de vital importancia. En su elección prevalecerá la decisión de los supervisores del proyecto y de la dirección de la empresa.

#### 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.

#### Tipos de entornos de desarrollo propietarios más relevantes en la actualidad.

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 profesional.	C/C++.	Windows.
JBuilder.	Java.	Windows, Linux, Mac OS X.
JCreator.	Java.	Windows.
Xcode.	C/C++, Java.	Mac OS X.

### Autoevaluación

Relaciona los siguientes entornos de desarrollo con sus características, escribiendo el número asociado a la característica en el hueco correspondiente.

Entorno de desarrollo.	Relación	Características.
Microsoft Visual Studio.	<input type="checkbox"/>	1. Libre. Soporta C/C++, Java, PHP, Javascript, Python.
NetBeans.	<input type="checkbox"/>	2. Propietario. Soporta Basic, C/C++, C#.
C++ Builder.	<input type="checkbox"/>	3. Propietario. Soporta C/C++.

## 4.- Estructura de entornos de desarrollo.

Los entornos de desarrollo, ya sean libres o propietarios, están formados por una serie de componentes software que determinan sus funciones.

Estos componentes son:

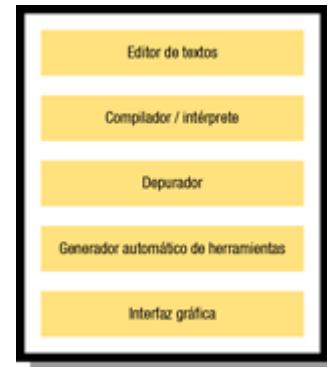
**Editor de textos:** Resalta y colorea la sintaxis, tiene la función de autocompletar código, ayuda y listado de parámetros de funciones y métodos de clase. Inserción automática de paréntesis, corchetes, tabulaciones y espaciados.

**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.



## 5.- Instalación de entornos integrados de desarrollo.

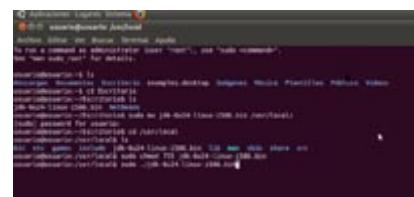
Vamos a realizar la instalación de NetBeans, en su versión 6.9.1 sobre Ubuntu 10.10. Tiene alguna complicación, porque se va a trabajar desde la terminal de Ubuntu. Te pedimos que prestes atención a los comandos.

### 5.1.- Instalación de JDK.

**La instalación del IDE NetBeans, ya sea en Linux, Windows o Mac OS X, requiere la instalación previa del JDK compatible con la versión de NetBeans que se quiera instalar.**

Lo primero es instalar el JDK en el sistema operativo. Esta será la plataforma del entorno, imprescindible para que éste pueda ser instalado en el sistema operativo y funcionar.

Se ha elegido como sistema operativo Linux. El proceso de instalación sólo podrá ser realizado por el root, que es el súper-usuario. Por ello, la instalación se realizará desde la consola de comandos:



Versión de JDK elegida:  
JDK-6u24-linux-i586.

Órdenes en la consola de comandos:

- ✓ Obtener el archivo, que se adjunta como recurso en la presente unidad.
- ✓ Mover el archivo a /usr/local.
- ✓ Darle permisos de ejecución, como root del sistema.
- ✓ Ejecutarlo, como root.

El proceso de instalación en Linux consta de una serie de pasos, y se explican con detalle en el siguiente documento:

**JDK son las siglas de Java Development Kit: Kit de desarrollo de Java. Consiste en la plataforma del entorno, imprescindible para que éste pueda ser instalado y ejecutado.**

## 5.2.- Instalación de NetBeans.

---

Una vez tenemos instalado el JDK en nuestro equipo, ya tenemos preparado el contexto en el que se instalará el entorno NetBeans.

La versión elegida es NetBeans 6.9.1. El archivo se puede descargar libremente desde el sitio web oficial y la instalación sólo puede ser realizada por el root. (Cuando estudies este módulo puede que haya una versión más reciente. De todas formas, es muy probable que las condiciones de instalación sean las mismas que las aquí descritas).

Recuerda repasar las recomendaciones de instalación que estarán en la página de NetBeans).

Eso nos fuerza a realizarla en la consola de comandos, y es un poco más compleja que en el caso del JDK.

Al igual que en el caso anterior, hay que darle al archivo permiso de ejecución y ejecutarlo.

El proceso de instalación se explica con detalle en el siguiente documento:

### Autoevaluación

**En tu opinión, ¿Por qué crees que la instalación del JDK sólo la puede realizar el root del sistema?**

- Porque se trata de un archivo binario de sistema.
- Porque ningún archivo puede ser ejecutado por un usuario que no sea el root.
- Porque estamos trabajando en la terminal del sistema.

## 6.- Configuración y personalización de entornos de desarrollo.

---

Una vez tenemos instalado nuestro entorno de desarrollo podemos acceder a personalizar su configuración.

Al abrir un proyecto existente, o bien crear un nuevo proyecto, seleccionaremos un desplegable con el nombre de “configuración” desde el que podremos personalizar distintas opciones del proyecto.

Podemos personalizar la configuración del entorno sólo para el proyecto actual, o bien para todos los proyectos, presentes y futuros.

Parámetros configurables del entorno:

- ✓ Carpeta o carpetas donde se alojarán todos los archivos de los proyectos (es importante la determinación de este parámetro, para tener una estructura de archivos ordenada).
- ✓ Carpetas de almacenamiento de paquetes fuente y paquetes prueba.
- ✓ Administración de la plataforma del entorno de desarrollo.
- ✓ Opciones de la compilación de los programas: compilar al grabar, generar información de depuración.
- ✓ Opciones de **empaqueñado** de la aplicación: nombre del archivo empaquetado (con extensión .jar, que es la extensión característica de este tipo de archivos empaquetados) y momento del empaquetado.
- ✓ Opciones de generación de documentación asociada al proyecto.
- ✓ Descripción de los proyectos, para una mejor localización de los mismos.
- ✓ Opciones globales de formato del editor: número de espaciados en las sangrías, color de errores de sintaxis, color de etiquetas, opción de autocompletado de código, propuestas de insertar automáticamente código.
- ✓ Opciones de combinación de teclas en teclado.
- ✓ Etc.

## 7.- Gestión de módulos.

---

Con la plataforma dada por un entorno de desarrollo como NetBeans podemos hacer uso de módulos y plugins para desarrollar aplicaciones.

En la página oficial de NetBeans encontramos una relación de módulos y plugins, divididos en categorías.

Seleccionando la categoría Lenguajes de Programación, encontraremos aquellos módulos y plugins que nos permitan añadir nuevos lenguajes soportados por nuestro IDE.

Un módulo es un componente software que contiene clases de Java que pueden interactuar con las API del entorno de desarrollo y el manifest file, que es un archivo especial que lo identifica como módulo.

Los módulos se pueden construir y desarrollar de forma independiente. Esto posibilita su reutilización y que las aplicaciones puedan ser construidas a través de la inserción de módulos con finalidades concretas. Por esta misma razón, una aplicación puede ser extendida mediante la adición de módulos nuevos que aumenten su funcionalidad.

Existen en la actualidad multitud de módulos y plugins disponibles para todas las versiones de los entornos de desarrollo más utilizados. En las secciones siguientes veremos dónde encontrar plugins y módulos para NetBeans 6.9.1 que sean de algún interés para nosotros y las distintas formas de instalarlos en nuestro entorno.

También aprenderemos a desinstalar o desactivar módulos y plugins cuando preveamos que no los vamos a utilizar más y cómo podemos estar totalmente actualizados sin salir del espacio de nuestro entorno.

Veremos las categorías de plugins disponibles, su funcionalidad, sus actualizaciones...

### Autoevaluación

**¿Cómo crees que influye el hecho de tener módulos y plugins disponibles en el éxito que tenga un IDE?**

- Contribuyen al éxito del entorno.
- No influyen en el éxito del entorno.

### 7.1.- Añadir.

---

Añadir un módulo va a provocar dotar de mayor funcionalidad a nuestros proyectos desarrollados en NetBeans.

Para añadir un nuevo módulo tenemos varias opciones:

1. Añadir algún módulo de los que NetBeans instala por defecto.
2. Descargar un módulo desde algún sitio web permitido y añadirlo.
3. Instalarlo on-line en el entorno.

Por supuesto, una cuarta posibilidad es crear el módulo nosotros mismos (aunque eso no lo veremos aquí).

Sin embargo, lo más usual es añadir los módulos o plugins que realmente nos interesan desde la web oficial de NetBeans. El plugin se descarga en formato .nbm que es el propio de los módulos en NetBeans. Posteriormente, desde nuestro IDE, cargaremos e instalaremos esos plugins. A esta manera de añadir módulos se le conoce como adición off-line.

También es habitual instalarlos on-line, sin salir del IDE.

La adición on-line requiere tener instalado el plugin Portal Update Center en NetBeans 6.9.1 y consiste en instalar complementos desde nuestro mismo IDE, sin tener que descargarlos previamente.

A modo de ejemplo, en el siguiente enlace tienes un documento donde se explican los pasos para añadir un módulo o plugin, de forma off-line (descargando el archivo e instalándolo posteriormente) y de forma on-line.

## 7.2.- Eliminar.

---

Cuando consideramos que algún módulo o plugin de los instalados no nos aporta ninguna utilidad, o bien que el objetivo para el cual se añadió ya ha finalizado, el módulo deja de tener sentido en nuestro entorno. Es entonces cuando nos planteamos eliminarlo.

Eliminar un módulo es una tarea trivial que requiere seguir los siguientes pasos:

1. Encontrar el módulo o plugin dentro de la lista de complementos instalados en el entorno.
2. A la hora de eliminarlo, tenemos dos opciones:
  1. Desactivarlo: El módulo o plugin sigue instalado, pero en estado inactivo (no aparece en el entorno).
  2. Desinstalarlo: El módulo o plugin se elimina físicamente del entorno de forma permanente.

Esta es la ventana, desde el gestor de complementos de NetBeans, que nos aparece cuando queremos eliminar un módulo del entorno.

Siempre nos pedirá elegir entre dos opciones: desactivar o desinstalar.

En este ejemplo, se opta por desactivar el complemento, como podemos ver en la imagen.

Para ver el ejemplo completo de desactivación de un complemento, se adjunta un breve documento con los pasos a seguir:

## Autoevaluación

**Para añadir un módulo desde la web oficial de NetBeans:**

- Hay que instalar el plugin Update Center.
- Hay que conectar con la web desde Netbeans y instalar on-line.
- Hay que encontrar el complemento, descargarlo y luego instalarlo en el IDE.
- No se pueden descargar los complementos desde ahí.

## 7.3.- Funcionalidades.

---

Los módulos y plugins disponibles para los entornos de desarrollo, en sus distintas versiones, tienen muchas y muy variadas funciones.

Podemos clasificar las distintas categorías de funcionalidades de módulos y plugins en los siguientes grupos:

1. **Construcción de código:** facilitan la labor de programación.
2. **Bases de datos:** ofrecen nuevas funcionalidades para el mantenimiento de las aplicaciones.
3. **Depuradores:** hacen más eficiente la depuración de programas.
4. **Aplicaciones:** añaden nuevas aplicaciones que nos pueden ser útiles.
5. **Edición:** hacen que los editores sean más precisos y más cómodos para el programador.
6. **Documentación de aplicaciones:** para generar documentación de los proyectos en la manera deseada.
7. **Interfaz gráfica de usuario:** para mejorar la forma de presentación de diversos aspectos del entorno al usuario.
8. **Lenguajes de programación y bibliotecas:** para poder programar bajo un Lenguaje de Programación que, en principio, no soporte la plataforma.
9. **Refactorización:** hacer pequeños cambios en el código para aumentar su legibilidad, sin alterar su función.
10. **Aplicaciones web:** para introducir aplicaciones web integradas en el entorno.
11. **Prueba:** para incorporar utilidades de pruebas al software.

## Autoevaluación

**¿Qué categoría de funcionalidad de NetBeans te parece más interesante? ¿Por qué?**

- Todas son igual de interesantes porque aumentan la funcionalidad.
- Depende de la tarea a realizar y el nivel del usuario.

## 7.4.- Herramientas concretas.

- ✓ Importador de Proyectos de NetBeans: permite trabajar en lenguajes como JBuilder.
- ✓ Servidor de aplicaciones GlassFish: Proporciona una plataforma completa para aplicaciones de tipo empresarial.
- ✓ Soporte para Java Enterprise Edition: Cumplimiento de estándares, facilidad de uso y la mejora de rendimiento hacen de NetBeans la mejor herramienta para crear aplicaciones de tipo empresarial de forma ágil y rápida.
- ✓ Facilidad de uso a lo largo de todas las etapas del ciclo de vida del software.
- ✓ NetBeans Swing GUI builder: simplifica mucho la creación de interfaces gráficos de usuarios en aplicaciones cliente y permite al usuario manejar diferentes aplicaciones sin salir del IDE.
- ✓ NetBeans Profiler: Permite ver de forma inmediata ver cómo de eficiente trabajará un trozo de software para los usuarios finales.
- ✓ El editor WSDL facilita a los programadores trabajar en servicios Web basados en XML.
- ✓ El editor XML Schema Editor permite refinar aspectos de los documentos XML de la misma manera que el editor WSDL revisa los servicios Web.
- ✓ Aseguramiento de la seguridad de los datos mediante el Sun Java System Acces Manager.
- ✓ Soporte beta de UML que cubre actividades como las clases, el comportamiento, la interacción y las secuencias.
- ✓ Soporte bidireccional, que permite sincronizar con rapidez los modelos de desarrollo con los cambios en el código conforme avanzamos por las etapas del ciclo de vida de la aplicación.
- ✓ Etc.



## Autoevaluación

**¿En qué fases del desarrollo de software ayudan los entornos integrados de desarrollo?**

- En codificación, pruebas, documentación, explotación y mantenimiento.
- En codificación y documentación.
- En análisis y documentación.

## 8.- Uso básico de entornos de desarrollo.

En el sitio principal del entorno de desarrollo de NetBeans nos encontramos con la siguiente ventana, que aparece cuando seleccionamos archivo, nuevo proyecto, java:

Vemos que el espacio se divide en dos ventanas principales.

- ✓ **Ventana Izquierda: ventana de proyectos.**  
Aquí irá apareciendo la relación de proyectos, archivos, módulos o clases que vayamos abriendo durante la sesión.
- Cada proyecto comprende una serie de archivos y bibliotecas que lo componen.
- El principal archivo del proyecto Java es el llamado **Main.java**.
- ✓ **Ventana derecha: espacio de escritura de los códigos de los proyectos.**  
Aquí aparece el esqueleto propio de un programa escrito en lenguaje Java.

Se ha añadido el código:

```
System.out.println("Hola Mundo");
```

Y veremos su significado en las siguientes páginas. De momento, saber que para escribir cualquier código, hay que hacerlo en esta ventana.

**BARRA DE HERRAMIENTAS:** Desde aquí podremos acceder a todas las opciones del IDE.

## 8.1.- Edición de programas.

---

En este sencillo ejemplo se ve una modificación de las líneas de código en la ventana de codificación del archivo **Main.java** del proyecto **ejemplo** que acabamos de crear.

Las dos líneas que aparecen resaltadas se han escrito sobre la ventana y, tal y como significan en lenguaje Java, su ejecución implicará que sendos mensajes encerrados entre comillas y entre paréntesis saldrán impresos.

No hay que decir que la programación en Java no es objeto del presente módulo, pero puedes probar con algunos ejemplos en Java que tengas de otros módulos.

Mientras escribimos en el editor de textos nos percatamos de varias características de NetBeans que ya hemos señalado en páginas anteriores:

- ✓ Autocompletado de código.
- ✓ Coloración de comandos.
- ✓ Subrayado en rojo cuando hay algún error y posibilidad de depuración y corrección de forma visual, mediante un pequeño ícono que aparece a la izquierda de la línea defectuosa.

## 8.2.- Generación de ejecutables.

---

Una vez tenemos el código plasmado en la ventana de comandos y libre de errores de sintaxis, los siguientes pasos son: compilación, depuración, ejecución.

Al ejecutar el ejemplo anterior, el resultado es:

Si a este ejemplo le añadimos la funcionalidad de JFrame, el resultado de la ejecución es:

Estos ejemplos aparecen detallados en el siguiente documento, al que accederás siguiendo el siguiente enlace:

## Autoevaluación

**Los pasos que debemos dar para generar un ejecutable son:**

- Ejecución directa.
- Ejecución, una vez que el editor esté libre de errores sintácticos.
- Una vez que el editor esté libre de errores, compilar, depurar y ejecutar.

## 9.- Actualización y mantenimiento de entornos de desarrollo.

El mantenimiento del entorno de desarrollo es una tarea fundamental que requiere tener todos sus componentes periódicamente actualizados.

También es de vital importancia realizar copias de seguridad sobre las bases de datos de nuestros proyectos por si ocurriera algún error o proceso defectuoso poder restaurarlos.

El mantenimiento y las actualizaciones se hacen de forma on-line. En NetBeans contamos con el complemento llamado Auto Update Services. Lo podemos encontrar en el siguiente enlace:

Una vez instalado, nos permitirá realizar continuas revisiones del entorno y actualizaciones de todos los plugins.



Para añadir módulos y plugins on-line, hay que tener este complemento instalado en el entorno.

La gestión de las bases de datos asociadas a nuestros proyectos es muy importante. Habrá que realizarles copias de seguridad periódicamente, para asegurar su restauración en caso de fallos en el sistema, y mantenerlas actualizadas para su posible portabilidad futura a nuevas versiones del entorno que utilicemos.

### Autoevaluación

**¿Cuál es la razón, en tu opinión, de que salgan nuevas versiones de los entornos de desarrollo tan rápidamente?**

- Para adaptarse a la evolución del hardware.
- Para incluir y modificar funcionalidades del entorno.

# Anexo I.- Instalación JDK en Ubuntu 10.10.

---

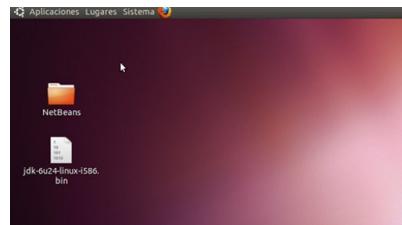
## PASOS:

### 1. Descargar el JDK de la siguiente URL:

[Descarga del JDK.](#)

El archivo de JDK utilizado es: **jdk-6u24-linux-i586.bin**

### 2. Guardar el archivo en el escritorio de Linux.



### 3. Mover el archivo al directorio /usr/local

El movimiento del archivo a esta ruta sólo puede ser realizado por el root del sistema.

Para poder ejecutarlo como un usuario normal, basta poner el comando **sudo** antes de la orden. Esto implica que todas las operaciones a partir de este momento deberemos realizarlas desde la terminal del sistema operativo.

Para acceder a la **terminal**, pulsamos sobre la pestaña de:

**Aplicaciones - Accesorios - Terminal**

Las acciones a realizar serán las siguientes:

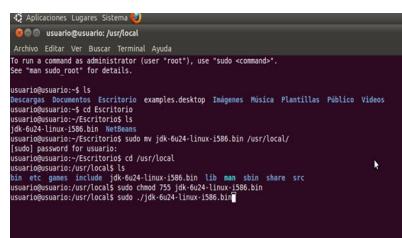
Entramos en el escritorio:

```
$ cd Escritorio
$ sudo mv jdk-6u24-linux-i586.bin /usr/local
```

### 4. Darle permiso de ejecución al archivo jdk y ejecutarlo

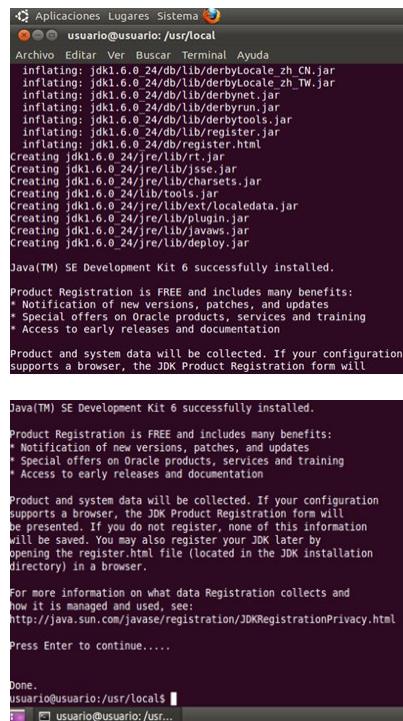
Entramos en la ruta:

```
$ cd /usr/local
$ sudo chmod 755 jdk-6u24-linux-i586.bin
$ sudo ./jdk-6u24-linux-i586.bin
```



Comienza la instalación...

## ED02\_Contenidos



```
Aplicaciones Lugares Sistema
Archivo Editar Ver Buscar Terminal Ayuda
inflating: jdk1.6.0_24/db/lib/derbyLocale_zh_CN.jar
inflating: jdk1.6.0_24/db/lib/derbyLocale_zh_TW.jar
inflating: jdk1.6.0_24/db/lib/derbyNet.jar
inflating: jdk1.6.0_24/db/lib/derbyRun.jar
inflating: jdk1.6.0_24/db/lib/derbyTools.jar
inflating: jdk1.6.0_24/db/lib/register.html
Creating jdk1.6.0_24/jre/lib/jse.jar
Creating jdk1.6.0_24/jre/lib/charsets.jar
Creating jdk1.6.0_24/jre/lib/tools.jar
Creating jdk1.6.0_24/jre/lib/ext/locatedata.jar
Creating jdk1.6.0_24/jre/lib/plugin.jar
Creating jdk1.6.0_24/jre/lib/javaws.jar
Creating jdk1.6.0_24/jre/lib/deploy.jar
Java(TM) SE Development Kit 6 successfully installed.

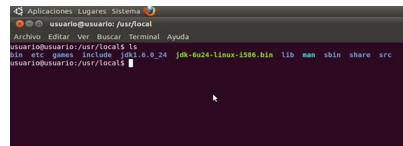
Product Registration is FREE and includes many benefits:
• Notification of new versions, patches, and updates
• Special offers on Oracle products, services and training
• Access to early releases and documentation

Product and system data will be collected. If your configuration
supports a browser, the JDK Product Registration form will
be presented. If you do not register, none of this information
will be saved. You may also register your JDK later by
opening the register.html file (located in the JDK installation
directory) in a browser.

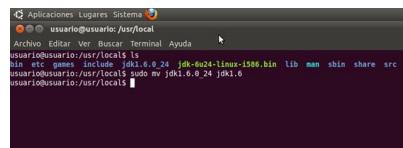
For more information on what data Registration collects and
how it is managed and used, see:
http://java.sun.com/javase/registration/JDKRegistrationPrivacy.html

Press Enter to continue.....
```

### 5. Renombramos la carpeta que se ha creado durante la instalación del archivo.

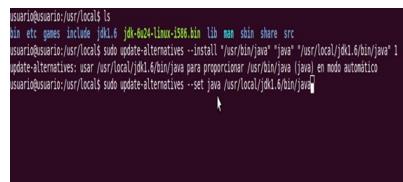


```
$ sudo mv jdk1.6.0_24 jdk1.6
```



### 6. Ejecutamos los siguientes comandos:

```
$ sudo update-alternatives --install "/usr/bin/java" "java" "/usr/local/jdk1.6/bin/jdk-6u24-linux-i586.bin" 1
$ sudo update-alternatives --set java /usr/local/jdk1.6/bin/java
```



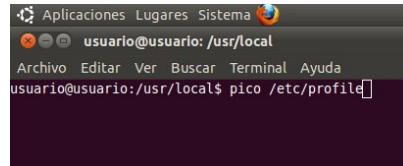
### 7. Editamos el archivo /etc/profile y agregamos las siguientes líneas al final del mismo:

```
export JAVA_HOME=/usr/local/jdk1.6
export PATH=$JAVA_HOME/bin:$PATH
```

Para editar el archivo podemos usar el comando:

```
$ pico /etc/profile
```

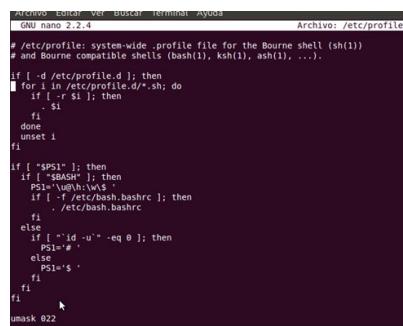
como se ve en la imagen:



o utilizar el comando:

```
$ nano /etc/profile
```

Cualquiera de los dos editores de texto (pico o nano) pueden ser usados en Linux. Despues de teclear cualquiera de los dos comandos anteriores nos aparece la siguiente ventana:

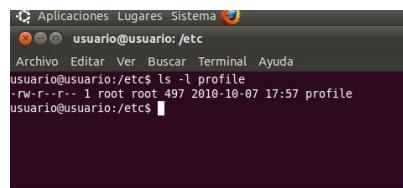


Nos colocamos al final del archivo y escribimos esas dos líneas:

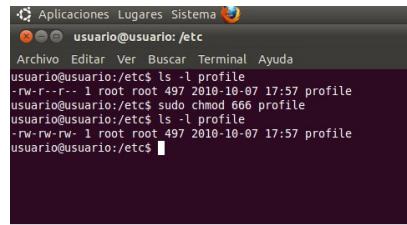
```
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).
if [ -d /etc/profile.d ]; then
    for i in /etc/profile.d/*; do
        if [ -r $i ]; then
            . $i
        done
        unset i
    fi
if [ "$PS1" ]; then
    if [ "$BASH" ]; then
        PS1="\u@\h:\w\$"
        if [ -f /etc/bash.bashrc ]; then
            . /etc/bash.bashrc
        else
            PS1="$ "
        fi
    fi
else
    if [ "$id" -u " -eq 0 ]; then
        PS1="# "
    else
        PS1="$ "
    fi
fi
fi
umask 022
export JAVA_HOME=/usr/local/jdk1.6
export PATH=$JAVA_HOME/bin:$PATH
```

Guardamos el archivo y nos dice que no tenemos permisos para modificarlo.

Por tanto, tenemos que darle a /etc/profile permiso de modificación:



## ED02\_Contenidos



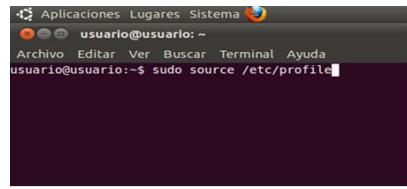
```
Aplicaciones Lugares Sistema
usuario@usuario:~/etc$ ls -l profile
-rw-r--r-- 1 root root 497 2010-10-07 17:57 profile
usuario@usuario:/etc$ sudo chmod 666 profile
usuario@usuario:/etc$ ls -l profile
-rw-rw-rw- 1 root root 497 2010-10-07 17:57 profile
usuario@usuario:/etc$
```

Ya sí podemos modificar el archivo agregándole las dos líneas al final del mismo (Repetir el paso de antes y guardar el archivo)

### 8. Salimos de la terminal, tecleando el comando exit, y volvemos a entrar en ella.

Teclear lo siguiente:

```
$ sudo source /etc/profile
```

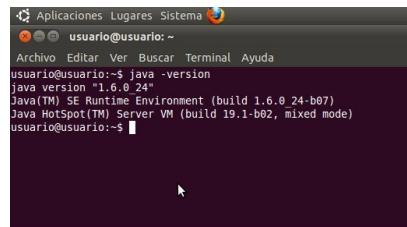


```
Aplicaciones Lugares Sistema
usuario@usuario:~$ sudo source /etc/profile
```

### 9. Probar el funcionamiento de Java.

Teclear el siguiente comando:

```
$ java -version
```



```
Aplicaciones Lugares Sistema
usuario@usuario:~$ java -version
java version "1.6.0_24"
Java(TM) SE Runtime Environment (build 1.6.0_24-b07)
Java HotSpot(TM) Server VM (build 19.1-b02, mixed mode)
usuario@usuario:$
```



## Anexo II.- Instalación NetBeans 6.9.1 en Ubuntu 10.10.

### PASOS:

1. Descargar NetBeans de la siguiente URL:

[Descargar Netbeans.](#)

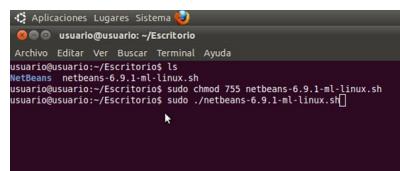
La versión de NetBeans utilizada es: **NetBeans 6.9.1** y el archivo de instalación es:

**Netbeans-6.9.1-m1-linux.sh**

2. Guardar el archivo en el escritorio de Linux.



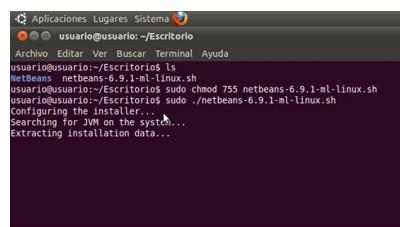
3. Le damos permiso de ejecución y lo instalamos



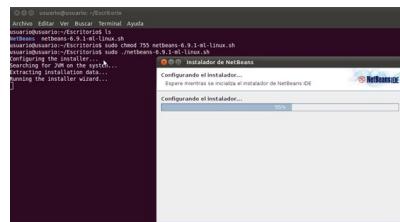
El código es:

```
$ sudo chmod 755 netbeans-6.9.1-m1-linux.sh
$ sudo ./netbeans-6.9.1-m1-linux.sh
```

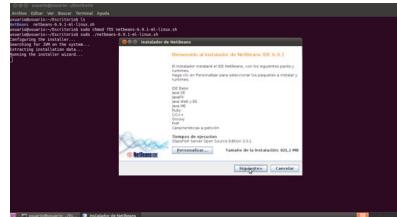
Comienza el proceso de instalación:



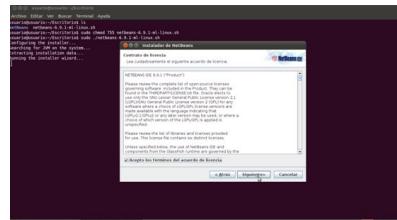
Durante la instalación nos aparecen sus imágenes gráficas correspondientes:



La instalación en sí es muy sencilla: basta con seleccionar "siguiente" en todas las opciones:

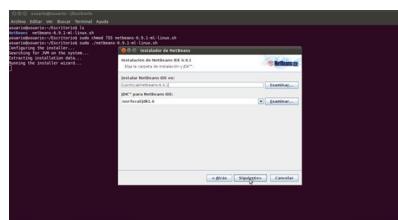


Aceptamos la licencia...

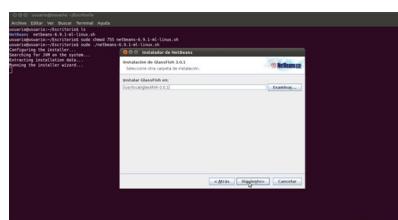


Podemos seleccionar el lugar donde se instalará NetBeans.

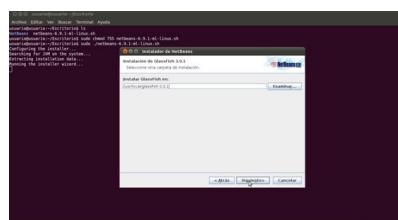
Aquí se ha dejado el lugar por defecto: seleccionamos siguiente.



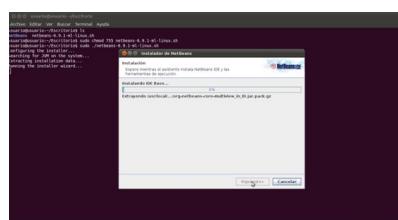
Volvemos a seleccionar siguiente:



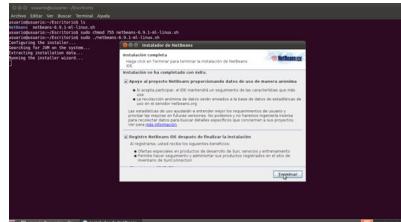
Terminada la fase de configuración previa, seleccionamos instalar:



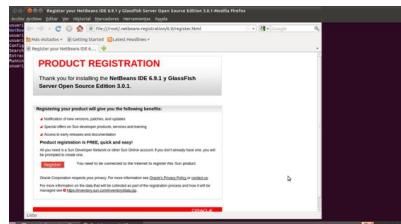
El proceso de instalación dura unos instantes...



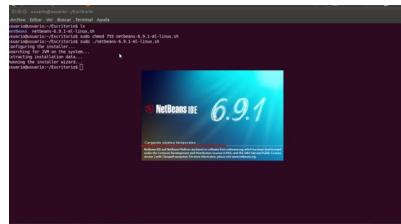
En la penúltima ventana de instalación nos preguntan si queremos registrarnos y colaborar con el proyecto de forma anónima. Esta elección es personal y aunque se ha seleccionado, no es necesario, aunque es una opción interesante.



Pulsamos sobre terminar.



Después de registrarnos (si así lo hemos querido), se abre la ventana de NetBeans:



Una opción muy útil de NetBeans es la incorporación de tutoriales on-line sobre los aspectos más destacados de este entorno de desarrollo:



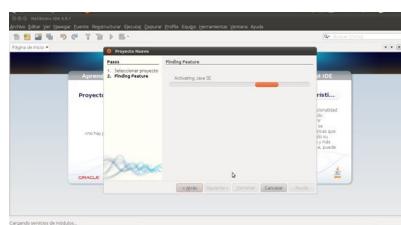
Si queremos hacer un nuevo proyecto, basta con seleccionar:

Archivo-Nuevo Proyecto

y aparece la siguiente ventana:

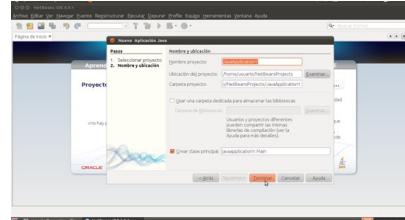


Seleccionamos la opción que nos interese y pulsamos sobre siguiente.

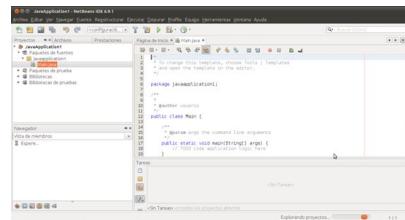


Le daríamos un nombre y la ubicación donde se va a guardar:

Finalmente, pulsamos Terminar.

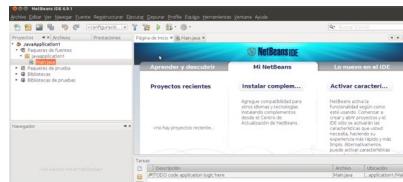


La apariencia primera del proyecto sería la siguiente:



## Anexo III.- Configuración y personalización de NetBeans.

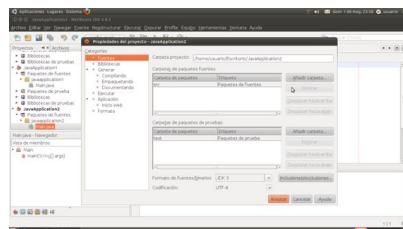
Accedemos a NetBeans y entramos en la página principal de la aplicación.



Para entrar a la aplicación podemos seleccionar "Nuevo Proyecto" y, una vez abierto, personalizar la configuración de NetBeans para ese proyecto.

En la barra de iconos de la aplicación, seleccionamos el desplegable de configuración

Seleccionamos "personalizar" y nos aparecerá la siguiente ventana:



Aquí vemos todo lo que podemos personalizar de la aplicación:

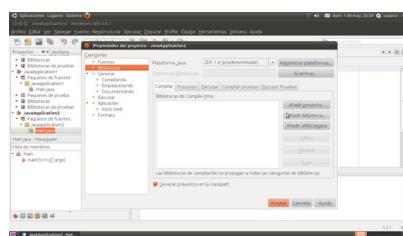
- ✓ Fuentes.
- ✓ Bibliotecas.
- ✓ Generación de código.
- ✓ Ejecución de código.
- ✓ Opciones de la aplicación.
- ✓ Formato del código en el editor de textos.

### FUENTES:

Podemos modificar:

- ✓ La carpeta que contendrá el proyecto
- ✓ La carpeta que almacenará los paquetes fuentes
- ✓ La carpeta que contendrá los paquetes prueba

### BIBLIOTECAS:



Desde esta ventana podemos elegir la plataforma de la aplicación.

Toma por defecto el JDK, pero se puede cambiar si se quiere, siempre y cuando sea compatible con la versión de NetBeans utilizada.

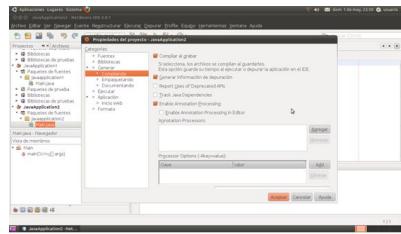
También en esta ventana se puede configurar el paquete de pruebas que se realizará al proyecto.

### GENERACIÓN DE CÓDIGO - COMPILANDO

Las opciones que nos permite modificar en cuanto a la compilación del programa son:

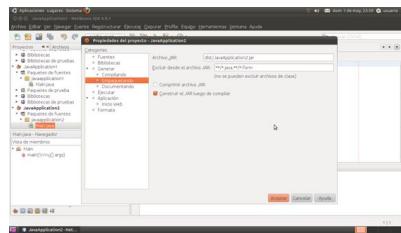
- ✓ Compilar al grabar: al guardar un archivo se compilará automáticamente.
- ✓ Generar información de depuración: para obtener la documentación asociada.
- ✓ Enable annotation processing: permitir anotaciones durante el proceso.

También podemos agregar anotaciones concretas para el proceso de compilación y añadir opciones de proceso que, según las características del proyecto, puedan ser de interés para nosotros.



## GENERACIÓN DE CÓDIGO - EMPAQUETANDO

Las aplicaciones resultado de la compilación del código deben ser empaquetadas antes de su distribución, con objeto de tener un único archivo, generalmente comprimido, que contenga en su interior todos los archivos de instalación y configuración necesarios para que la aplicación pueda ser instalada y desarrollada con éxito por el usuario cliente.

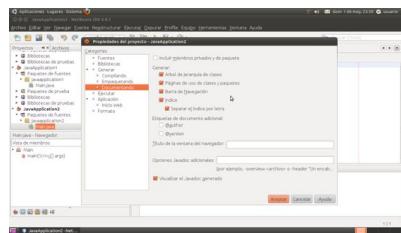


Como vemos en la imagen, en esta opción podemos modificar el lugar donde se generará el archivo resultante del empaquetado, así como si deseamos comprimirlo.

También podemos elegir que el archivo empaquetado se construya tras la compilación, que es lo habitual (por eso esta opción aparece como predeterminada)

## GENERACIÓN DE CÓDIGO - DOCUMENTANDO

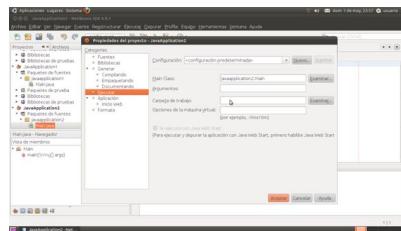
Como ya vimos en la unidad anterior, la documentación de aplicaciones es un aspecto clave que no debemos descuidar nunca. NatBeans nos ofrece una ventaja muy considerable al permitirnos obtener documentación de la fase de codificación de los programas de forma automática.



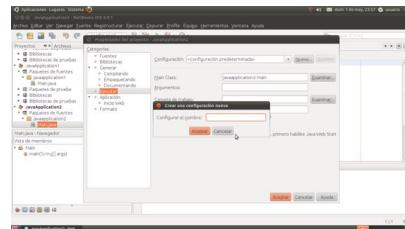
Dentro del documento que se va a generar podemos elegir que se incluyan todas las opciones anteriores. Esto es lo más recomendable, por eso aparecen todas marcadas de forma predeterminada y lo mejor es dejarlo como está.

## EJECUTANDO CÓDIGO

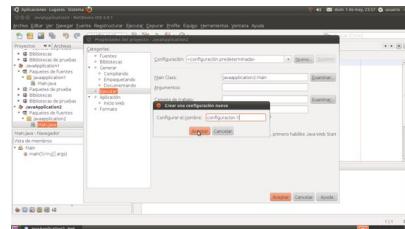
Esta opción nos permite definir una nueva configuración de ejecución de código, elegir la clase principal, las carpetas de trabajo del proyecto y opciones de la máquina virtual.



## ED02\_Contenidos

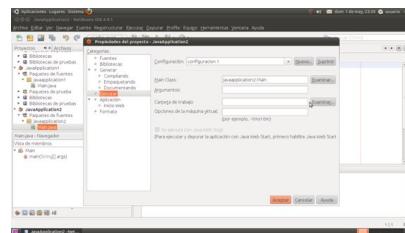


En la ventana de “Configurar el nombre” escribimos el nombre que tendrá nuestra configuración personalizada.

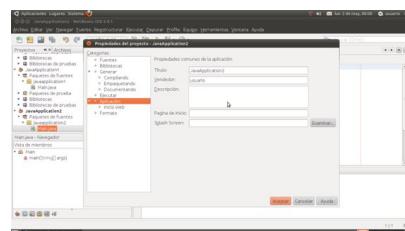


En este caso, escribimos “configuración 1” y pulsamos “aceptar”

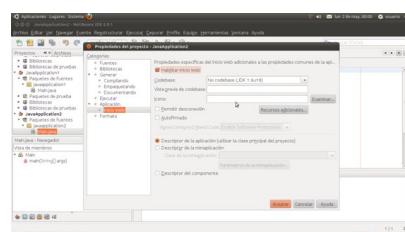
A partir de este momento, todas las opciones de configuración que seleccionemos que guardarán en “configuración 1”



Ahora podemos elegir la aplicación sobre la cual queremos aplicar la configuración personalizada de “configuración 1”



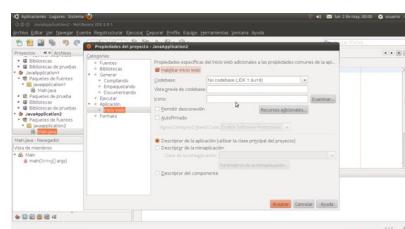
### OPCIONES DE LA APLICACIÓN



Como vemos, podemos dar una descripción al proyecto, cambiarle el nombre, etc...

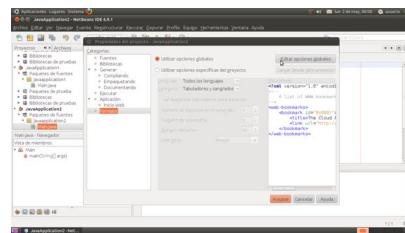
Es conveniente hacerlo, ya que el nombre de los nuevos proyectos se generar automáticamente por NetBeans al inicio de la sesión.

En cuanto las opciones del inicio web:

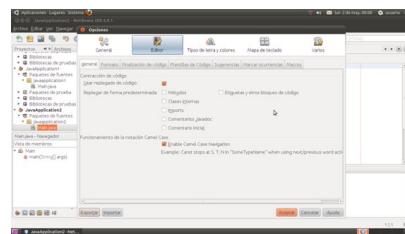


## FORMATO

Aquí podemos personalizar aspectos globales del formato del código fuente en la aplicación. Podemos personalizar las opciones sólo para el proyecto actual o bien para todos los proyectos que estén basados en NetBeans a partir de ahora (utilizar opciones globales)

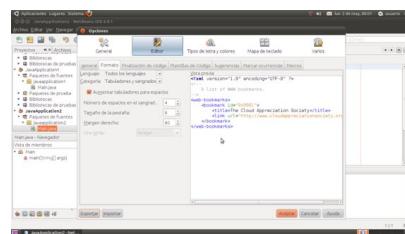


Si seleccionamos Editar opciones globales nos encontramos con la siguiente ventana, que tiene una barra superior de pestañas para configurar cada apartado del formato de forma independiente:

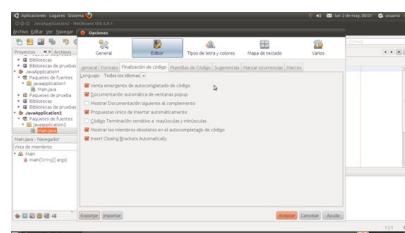


Pestaña Formato:

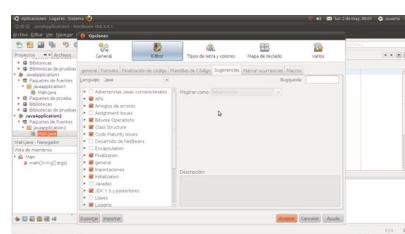
Se puede configurar los tamaños de los espaciados, pestañas, etc...



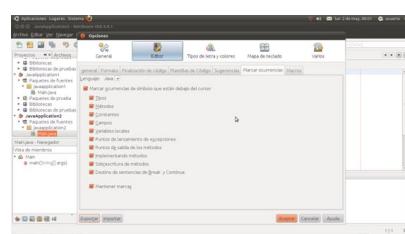
En la pestaña de Finalización de código:



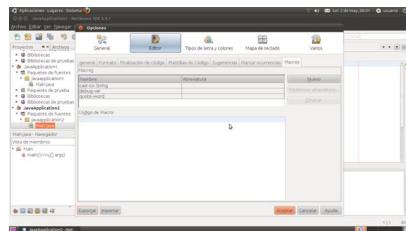
En la pestaña de sugerencias:



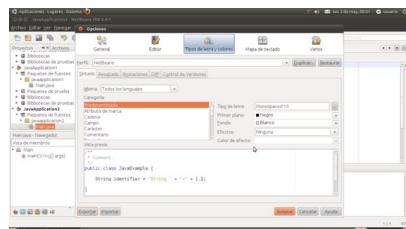
En la pestaña de Marcar ocurrencias:



En la pestaña de macros:

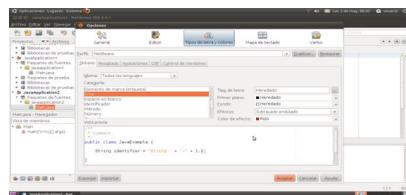


En cuanto al ícono de Tipos de letra y colores:



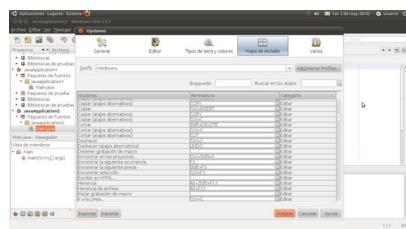
Consiste en elegir el tipo de letra y colores que prefiramos para el texto del código, así como efectos (si es que así lo deseamos)

También podemos configurar el tipo de letra y color de los errores del código (por defecto, de color rojo)



Y lo mismo con los números, espacios en blanco, etc...

En cuanto a los métodos abreviados de teclado (combinación de teclas equivalente a las acciones en NetBeans), podemos modificar aquellas acciones que hagamos con más frecuencia por aquella combinación de teclas que nos sea más fácil recordar.



## Anexo IV.- Adición de módulo en NetBeans.

Hay dos formas de añadir módulos y plugins en NetBeans:

1. Off-line: Buscar y descargar plugins desde la página web oficial de la plataforma:

[Descarga de plugins para NetBeans.](#)

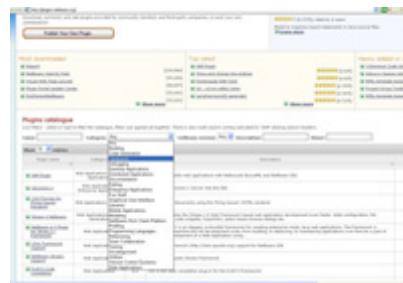
### Ejemplo:

Vamos a buscar un plugin para jugar al sudoku desde nuestro IDE. No es muy educativo, pero sirva como ejemplo la manera en que se va a realizar el proceso (será igual en todos los casos):

Entramos en la zona de descargas de plugins para NetBeans y en la zona del catálogo, escribiremos la palabra sudoku:



sudoku



Se nos abre una ventana con las características del plugin y la opción de descargarlo. Elegimos la carpeta donde queramos que se guarde.

Entramos en NetBeans:



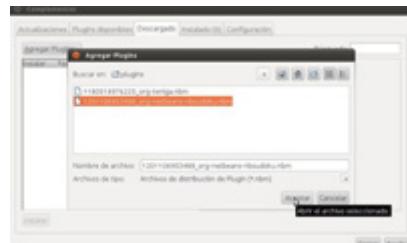
Creamos nuevo proyecto y seleccionamos el tipo de proyecto que queramos (por ejemplo, aplicación de Java).

Herramientas - Complementos:

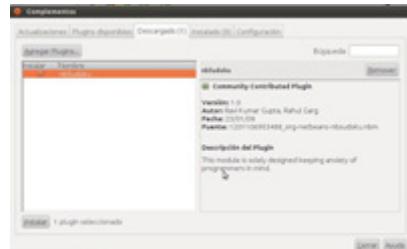


En la pestaña "descargado" seleccionamos "Agregar Plugins"

## ED02\_Contenidos



Seleccionamos la carpeta donde habíamos guardado el plugin del sudoku y le damos a "aceptar"



Estando el plugin seleccionado, pulsamos "instalar".

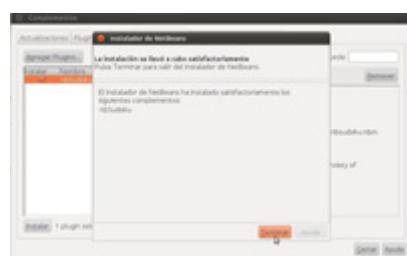
Empieza la instalación:



Pulsamos siguiente. Despues, aceptamos la licencia:

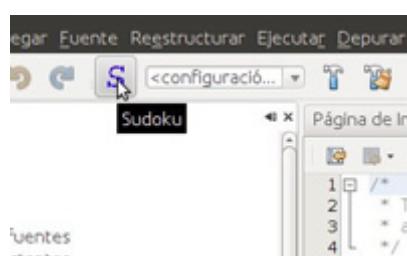


Pulsamos "instalar"



Seleccionamos "Terminar"

Observamos el icono que aparece en la barra de iconos superior del sitio:



Si lo pulsamos, ya podemos jugar un ratito al sudoku para despejarnos:

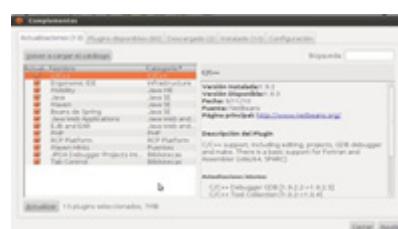


## 2. On-Line: Instalarlos desde el propio entorno de desarrollo:

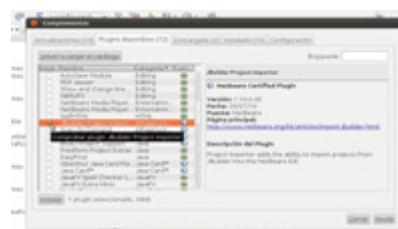
Ahora vamos a instalar otros plugins con mayores utilidades que el anterior... vamos a hacer dos ejemplos instalando dos plugins diferentes:

- ✓ **Pdf Viewer:** Nos permitirá abrir archivos en pdf desde el propio IDE, emergiendo una nueva ventana en el sitio específica para ello.
- ✓ **Importador de bibliotecas y proyectos de JBuilder.**

Estando en NetBeans, seleccionamos **Herramientas - Complementos:**



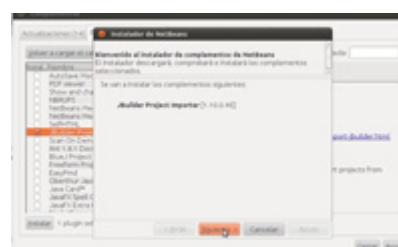
En la pestaña de plugins disponibles: seleccionamos **JBuilder - Instalar**



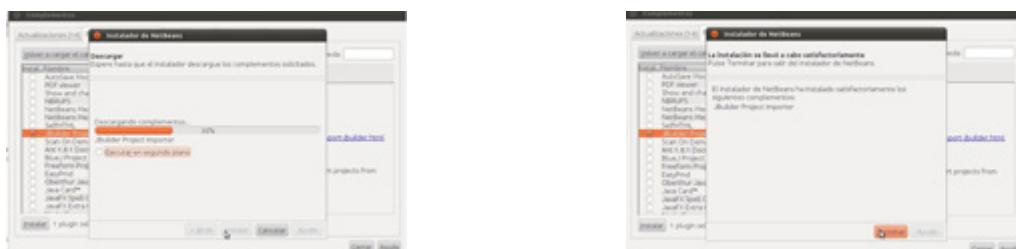
Se abre la siguiente ventana:



Aceptamos los términos de la licencia y pulsamos sobre **Instalar**.



Pulsamos siguiente



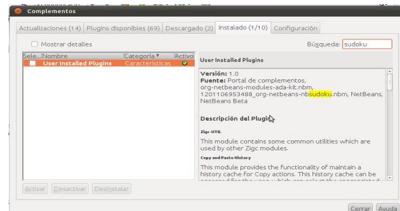
Pulsamos sobre **Terminar.**

## Anexo V.- Eliminar módulos en NetBeans.

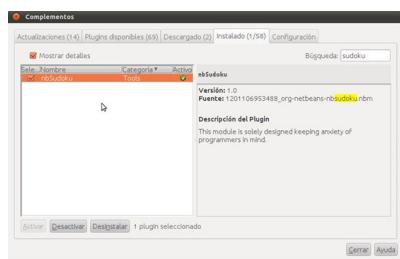
Vamos a ver la secuencia de pasos a seguir para eliminar el plugin del juego del sudoku del entorno.

El proceso es muy sencillo: basta con conseguir la lista de complementos instalados (Herramientas - Complementos). Localizamos el complemento que queremos eliminar escribiendo su nombre en el lugar destinado para ello y seleccionamos una de entre las dos opciones posibles: desinstalarlo o desactivarlo

En la pestaña de complementos instalados, escribimos el nombre del plugin (sudoku) en la barra de búsqueda:



Cuando lo encuentra, en la ventana aparecen las dos posibilidades de eliminación:



En este caso, hemos optado por desactivarlo.



## Anexo VI.- Ejemplo de edición de código.

En este documento vamos a introducirnos en la edición de programas en NetBeans a través de un ejemplo sencillo de una aplicación de Java.

Lo primero es iniciar la plataforma:

Seleccionamos archivo - nuevo proyecto.

Elegimos una aplicación de Java:

Lo vamos a llamar ejemplo.

Una vez iniciado el proyecto, en la ventana de proyectos (izquierda) vemos cómo se ha cargado el proyecto ejemplo. Lo seleccionamos con el ratón y se despliega, mostrando todos sus archivos componentes. Seleccionamos Main.java (que es el archivo principal del proyecto, el cual vamos a editar):

En la ventana de edición (a la derecha) nos aparece el esqueleto de la estructura básica de una aplicación en Java.

Lo que vamos a hacer a lo largo del ejemplo es añadir código.

La primera línea de código que vamos a agregar es una orden sencilla en Java, cuya ejecución posterior dará lugar a la aparición de un mensaje por pantalla.

La apariencia del IDE será la siguiente:

Añadimos otra línea más con otro mensaje "Creando mi primer ejemplo"

Ahora vamos a modificar la parte de arriba del programa. Añadimos la siguiente línea:

Esta línea nos va a servir para adentrarnos en una de las utilidades más importantes de NetBeans 6.9.1.

NetBeans entiende esta orden como un error (aparece subrayada en una línea roja ondulada y con un pequeño ícono al lado izquierdo)

Si pulsamos sobre ese ícono con el ratón, NetBeans nos aporta sugerencias para deshacer el error:

En este caso, elegimos importar JFrame a la librería.

Y seguimos añadiendo código en el editor:

Se nos vuelven a subrayar líneas en rojo, actuamos igual que en el caso anterior y vamos viendo las sugerencias que nos dan para corregir. También vamos viendo las opciones de autocompletado de código:

Llegados a este punto, ya hemos comprobado que el editor no nos da ningún problema más. En el siguiente punto del tema, veremos cómo ejecutar esto.

Vemos también cómo se han importado con éxito las librerías que nos han hecho falta:

El código completo del ejemplo es el siguiente:

```
package ejemplo;

import javax.swing.JFrame;
import javax.swing.JLabel;
public class Main
extends JFrame {

    public Main() {
        JLabel lblSaludo = new JLabel( "Hola Mundo. Creando mi primer ejemplo")
        add(lblSaludo);
```

## Anexo VII.- Ejecución de un programa en NetBeans.

---

Continuando con el ejemplo anterior, recuerda que habíamos llegado a este punto:

Tenemos el programa escrito en el editor libre de errores sintácticos.

¿Cómo convertir ese programa en ejecutable?

Cabe destacar que, por la sencillez y pequeñez del programa, la ejecución del mismo podría ser directa sin ningún problema.

Sin embargo, debemos acostumbrarnos a seguir los pasos adecuados, que son:

- ✓ Editor libre de errores → Compilación → Depuración → Ejecución
- ✓ Para compilar un programa, debemos seleccionar ejecutar (en la barra superior de herramientas) → Compile File
- ✓ Depurar → Barra de herramientas
- ✓ Ejecutar → En la barra de herramientas o bien mediante el ícono de acceso directo en la parte superior de la ventana de edición de código.