

MP_0487. Entornos de desarrollo

**UF2. Instalación y explotación de las
herramientas de desarrollo**

**2.2. Funciones de un
entorno de desarrollo**

Índice

☰	Objetivos	3
☰	Editores orientados a lenguajes de programación	4
☰	Compiladores y enlazadores	6
☰	Generadores de programas	8
☰	Depuradores	10
☰	Prueba y validación software	12
☰	Optimizadores de código	14
☰	Empaquetadores	15
☰	Generadores de documentación	17
☰	Gestores y repositorios de paquetes	19
☰	Control de versiones	20
☰	Creación de proyectos y primeros pasos	21
☰	Ayuda y documentación	22
☰	Configuración y personalización de Eclipse	23
☰	Asistentes de código	24
☰	Generación de código	25
☰	Resumen	26

Objetivos

En esta lección perseguimos los siguientes objetivos:

1

Describir las funcionalidades de las herramientas de uso común en el desarrollo software, así como enumerar los productos más populares para cada tipo.

2

Conocer distintos tipos de herramientas que nos permitirán editar, compilar, depurar, testar, empaquetar y generar programas.

3

También veremos programas que nos permitan generar documentación, gestionar repositorios y realizar el control de versiones.

4

Conoceremos los IDE más populares, que nos permiten realizar este tipo de gestiones.

¡Ánimo y adelante!

Editores orientados a lenguajes de programación

Un editor es un programa ligero que permite la creación y edición de ficheros de texto, pero con determinadas características que ayudan a los programadores a escribir los códigos en un determinado lenguaje.

Por ejemplo, con el Bloc de notas de Windows (*notepad*) sería suficiente para escribir el código de un programa, pero el programador no tendría ninguna facilidad para ello.

Existen muchas alternativas gratuitas que facilitan el trabajo de los programadores por las características que ofrecen, como:

- Coloreado de la sintaxis.
- Edición de múltiples ficheros.
- Formateo de texto.
- Conexiones con repositorios o FTP.
- Enlaces con otras herramientas.

De entre los editores orientados a lenguajes de programación más populares podemos mencionar dos:

SUBLIME TEXT

Para Linux, Windows y Mac. Destaca por lo ligero y simple que es, el aspecto visual sencillo pero elegante, los automatismos para realizar cambios de código, pero sobre todo por su gran capacidad de personalización.

NOTEPAD ++

Para Windows. Está escrito en C++ y es sumamente ligero. Cuenta con resaltado de sintaxis, búsqueda, interfaz personalizable, mapa del documento, autocompletado, pestañas para abrir múltiples documentos al mismo tiempo, etc.

Existen muchos más, como Light Table, Vim, Brackets, EditRocket y Emacs.

Compiladores y enlazadores

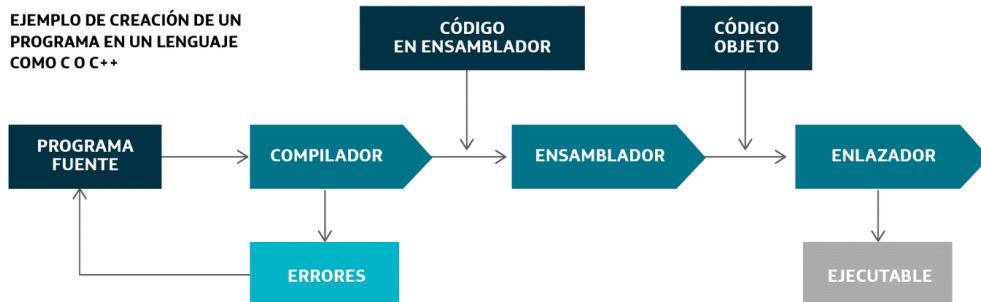
La creación de un programa ejecutable normalmente sigue dos pasos:

Compilación

En este paso el código fuente (escrito en un determinado lenguaje de programación) es traducido a código objeto o de bajo nivel.

Enlazado

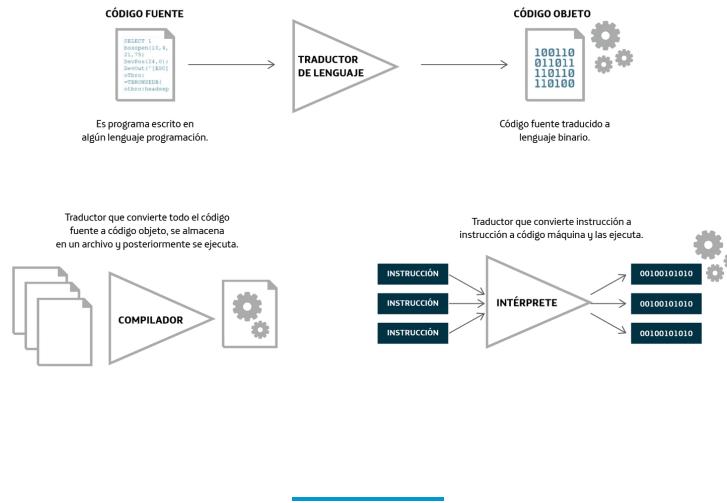
Este segundo paso traduce el código objeto a código máquina y genera un fichero ejecutable o una biblioteca.



Para obtener el código objeto que se debe ejecutar en una determinada máquina (ordenador) se utilizan los “traductores”.

Básicamente existen dos tipos de “traductores”:

- Los compiladores.
- Los intérpretes.



El trabajo del compilador se divide en tres fases:

Fase de análisis

- **Análisis léxico:** se lee el código fuente y se agrupa por componentes léxicos. Se usan autómatas finitos que acepten expresiones regulares para identificar los *tokens*.
- **Análisis sintáctico:** en esta fase se agrupan los *tokens* por jerarquías, generando un árbol de análisis sintáctico.
- **Análisis semántico:** en esta fase se revisa el programa fuente con el objetivo de encontrar posibles errores semánticos y para reunir información sobre los tipos para la posterior fase de generación de código.

Fase de síntesis

En esta fase se genera el código objeto del programa fuente. Este paso solo se hace cuando el código no tiene errores de análisis. En esta fase se produce la generación de código intermedio.

Optimización de código

Busca la mejora del código intermedio (compiladores optimizadores).

Generadores de programas

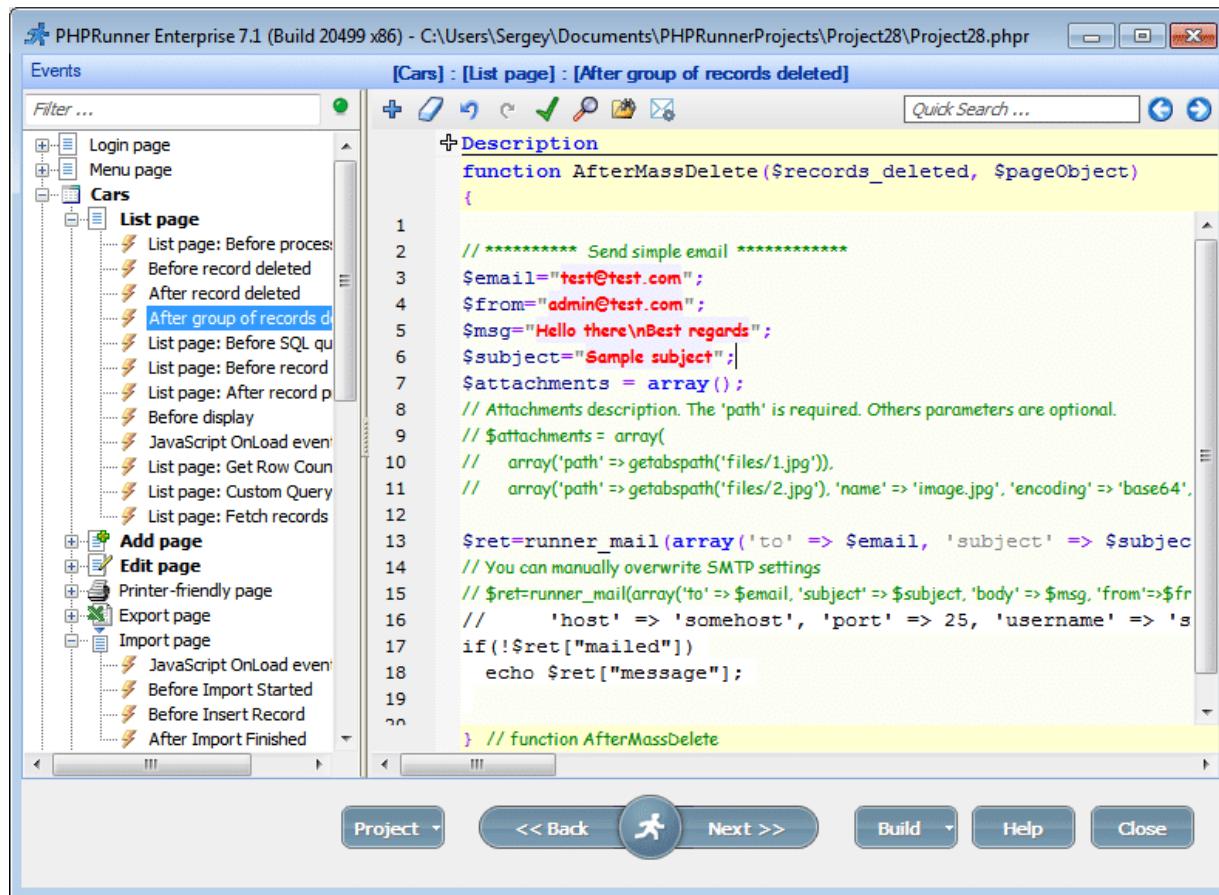
La programación automática pretende que sea el propio ordenador el que genere los programas que necesitamos, siguiendo unas pautas indicadas previamente (esta vez sí) por un programador.

Tenemos muchos ejemplos de programas que escriben otros programas. Tal es el caso por ejemplo de los **traductores de lenguajes** ya vistos, los cuales toman un programa fuente y realizan una serie de transformaciones hasta producir un ejecutable.

Muchos **IDE (entornos de desarrollo)** generan código de forma automática. Se elige el tipo de proyecto a desarrollar y el IDE crea la estructura de ficheros con códigos básicos para que el programador los modifique y amplíe.

Incluso cuando se trabaja con interfaces gráficas que, por ejemplo, para añadir un botón se realiza arrastrándolo con el ratón al lugar deseado, esta herramienta genera automáticamente el código tanto de su ubicación como de sus eventos asociados.

También las herramientas de “**análisis y diseño**” permiten la generación de los ficheros fuente de las clases que forman la jerarquía a partir de sus diagramas. Los programadores tienen que completar su código.



PHPRunner genera una interfaz totalmente funcional en PHP sobre una base de datos, a partir de la misma.

Pantalla PHPRunner - <https://xlinesoft.com/phprunner>.

Depuradores

Los IDE (entornos de desarrollo) tienen integrado su propio depurador.

Los **depuradores** o *debugger* son programas que se encargan de ayudar a eliminar los errores de programación. También tienen funciones de ejecución “paso a paso” así como de pausar la ejecución mediante un punto de ruptura (*breakpoint*) bajo ciertas condiciones. Durante esa interrupción el usuario puede examinar la pila de llamadas, el contenido de los registros, cambiar el punto de ejecución, etc.

Distinguimos dos tipos de depuradores:

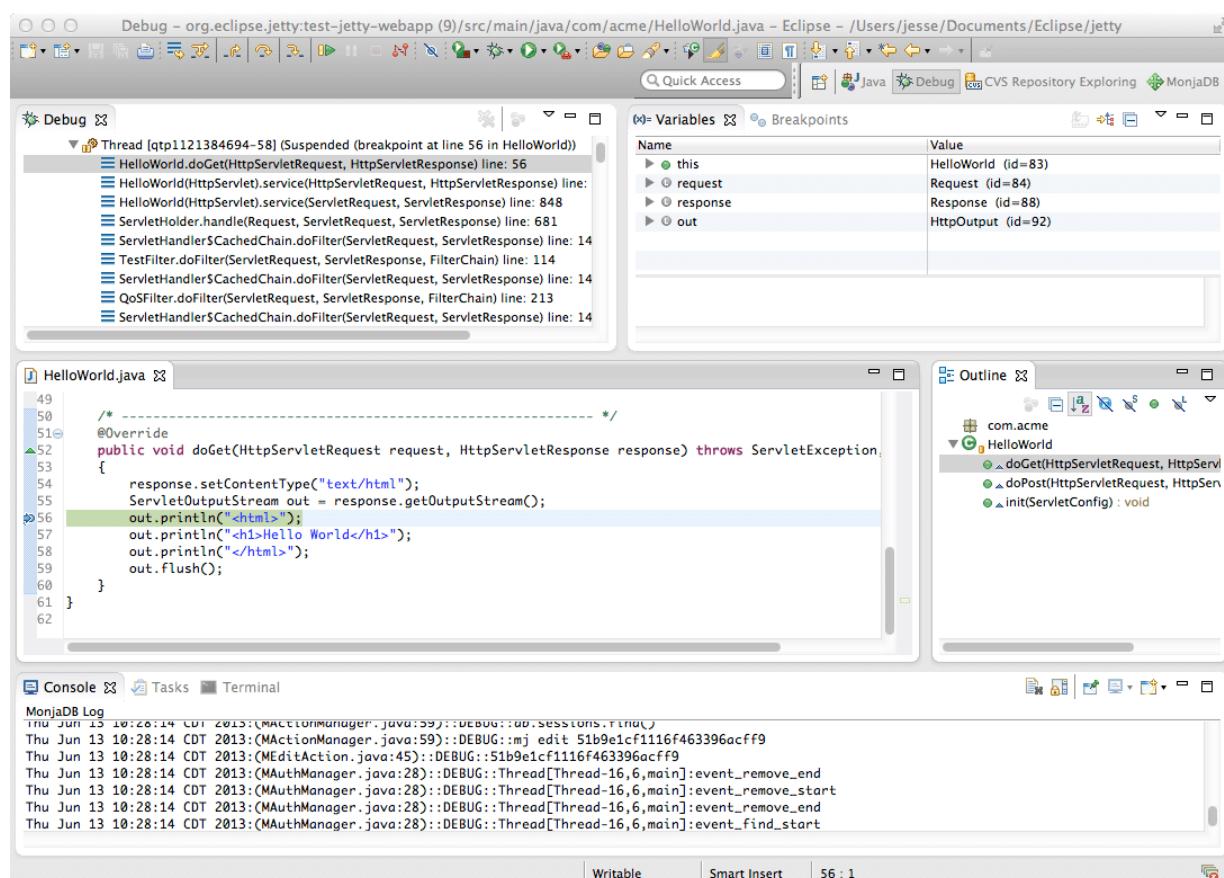
Depurador simbólico o de nivel fuente

En el momento en que el programa tiene un error o alcanza una condición predefinida, se muestra la posición en el código.

Depurador de lenguaje máquina o bajo nivel

Muestra la línea en el código fuente desensamblado.

Algun ejemplo de depuradores: GNU Debugger, Visual DuxDebugger, SoftICE y OllyDbg.



Debug Eclipse.

En un IDE, como por ejemplo Eclipse, se pueden realizar operaciones de depuración con las utilidades de "Debugger" incorporadas:

- *Breakpoints*.
- Menús.
- Barras de herramientas.
- Ventanas de inspección.
- Etc.

Prueba y validación software

Como ya sabemos, la validación software se realiza para confirmar que un software está en condiciones para satisfacer las necesidades del usuario.

Hay dos tipos de pruebas

Pruebas de defectos

Buscan los errores en el código, demostrando la presencia de defectos.

Pruebas estadísticas

Busca demostrar que el programa cumple la especificación y que es fiable.

Ejemplos de herramientas de pruebas de validación pueden ser las que encontramos dentro de **xUnit**. En esta familia de entornos de unidad de prueba se encuentran JUnit (para Java), CppUnit (C++), NUnit (.NET, C #, J #, C++ o VB.Net) o PyUnit (Python).

```
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assumptions.assertTrue;
import static org.junit.jupiter.api.Assumptions.assumingThat;

import org.junit.jupiter.api.Test;

class AssumptionsDemo {

    @Test
    void testOnlyOnCiServer() {
        assertTrue("CI".equals(System.getenv("ENV")));
        // remainder of test
    }

    @Test
    void testOnlyOnDeveloperWorkstation() {
        assertTrue("DEV".equals(System.getenv("ENV")),
            () -> "Aborting test: not on developer workstation");
        // remainder of test
    }

    @Test
    void testInAllEnvironments() {
        assumingThat("CI".equals(System.getenv("ENV")),
            () -> {
                // perform these assertions only on the CI server
                assertEquals(2, 2);
            });
        // perform these assertions in all environments
        assertEquals("a string", "a string");
    }

}
```

Ejemplo de implementación de pruebas con JUnit para una clase Java.

Optimizadores de código

Su **objetivo** es conseguir un código más eficiente en tiempo de ejecución (optimización temporal) y en espacio de memoria (optimización espacial).

La optimización de código se realiza después de que se hayan generado las líneas de todo el programa o de un elemento ejecutable del mismo.

El proceso que llevan a cabo para su objetivo es diverso, por ejemplo:

- Reordenación de código.
- Reducción de tamaño.
- Optimización de bucles.
- Eliminación de redundancias.
- Reordenación de operaciones.
- Eliminación de asignaciones no utilizadas.



El optimizador de código forma parte del compilador y se ejecuta automáticamente, los programadores normalmente no tienen que realizar ninguna acción.

Empaquetadores

Los empaquetadores son pequeños programas que se utilizan para recoger una serie de prioridades del futuro usuario para **realizar una instalación de acuerdo a sus necesidades.**

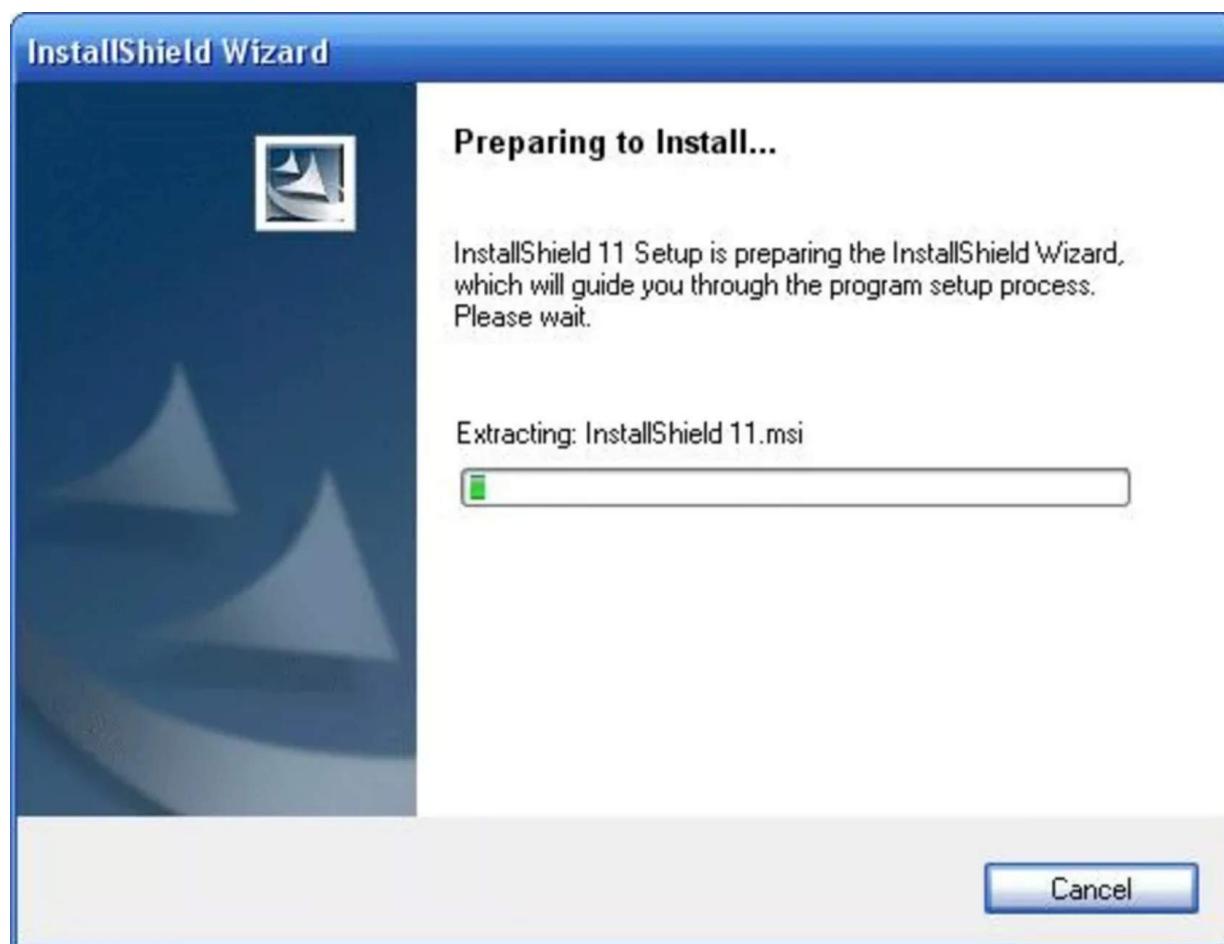
Gracias al uso de estas herramientas se puede hacer un “setup” que permita que el usuario haga la instalación de la aplicación desarrollada de acuerdo a sus necesidades y adapta el software al hardware.

InnoSetup

Una de las mejores herramientas gratuitas. Con soporte para todas las versiones de Windows, permite crear un único archivo .exe que contenga toda la aplicación. También permite la desinstalación.

InstallShield

Para instalar software de escritorio y las plataformas de servidor de Microsoft Windows. También se puede usar para administrar aplicaciones y paquetes de software en una amplia gama de dispositivos móviles y portátiles.



Generadores de documentación

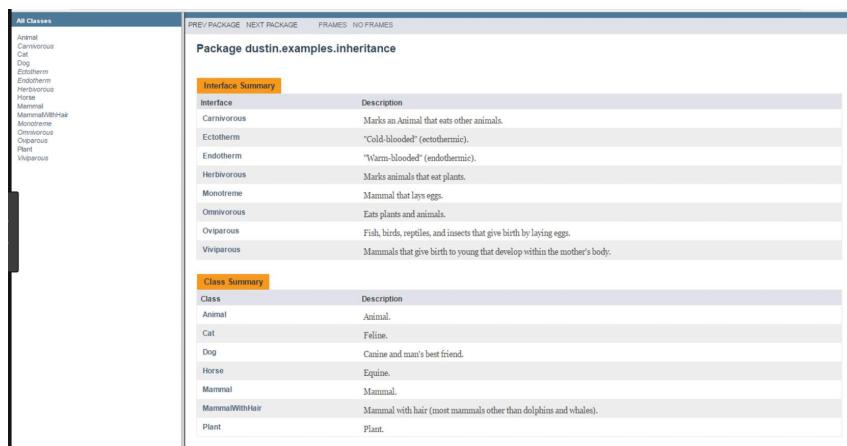
Un generador de documentación es una **herramienta software que genera de manera automática documentación** a partir de un código fuente (y a veces también a partir de código binario).

Esta documentación creada puede estar dirigida a los programadores (API) y/o a usuarios finales.

Tipos de documentos:

- Documentos automatizados o *batch*.
- Documentos interactivos.
- Formularios.

Ejemplos: Javadoc, Doxygen, Epydoc, JSDocf, PHPDocumentor, YARD, etc.



Ejemplo Javadoc.

Con Javadoc podemos generar documentación Java utilizando HTML a partir de comentarios en los archivos del código fuente para las clases almacenadas en un paquete.

Algunas etiquetas en los comentarios son:

- **@author:** nombre del programador.
- **@param:** descripción de cada uno de los parámetros.
- **@return:** lo que devuelve la función.
- **@version:** versión.

Gestores y repositorios de paquetes

Un sistema de gestión de paquetes es un conjunto de herramientas que se utilizan para que la **instalación, actualización, configuración y eliminación** de los paquetes software se realice de manera automática.

El sistema de gestión de paquetes forma parte del sistema operativo y puede administrar y verificar todos los paquetes del mismo con el objetivo de mantener su usabilidad.

Los gestores de paquetes **ayudan a resolver las dependencias software**, ya que estos, cada vez que se instala software nuevo, se encargan de calcular las dependencias y las descargan.

Uno de los gestores de paquetes más populares en Linux es **Aptitude**, usado en las distribuciones de Debian y Ubuntu.

Windows 10 incluirá un gestor de paquetes para la instalación y actualización de aplicaciones de forma sencilla y segura llamado **OneGet**, que podrá instalar programas solo con un comando en **PowerShell**, la consola avanzada de Windows.

Control de versiones

Los sistemas de control de versiones o SVC se encargan de gestionar de una manera ágil aquellos cambios que se realizan durante el desarrollo de un producto.

Este tipo de herramientas se utilizan mucho para controlar las numerosas versiones del código fuente o SCM (*Source Code Management*, en español administración de código fuente). Cada miembro del equipo de desarrollo debe crearse una copia local desde el repositorio. Sobre dicha copia realizará el trabajo pertinente.

Las modificaciones del código principal se pueden gestionar de forma exclusiva (se indica al repositorio qué elemento se desea cambiar e impide durante la actualización que otros usuarios lo modifiquen) o de forma colaborativa (cada uno modifica su copia local y al compartir los cambios es la herramienta la que trata de combinar las modificaciones).

Ejemplos de este tipo de herramientas son: CVS, SourceSafe, Subversion, Git, Darcs, Bazaar, Visual SourceSafe.

Creación de proyectos y primeros pasos



En [este vídeo](#) te mostramos los primeros pasos para crear un proyecto en Eclipse.

Ayuda y documentación



En [este vídeo](#) te mostramos cómo utilizar la ayuda de Eclipse.

Configuración y personalización de Eclipse



En [este vídeo](#) te mostramos las opciones de personalización y configuración de Eclipse.

Asistentes de código



Seguimos mostrándote funcionalidades de Eclipse; veamos ahora cómo [usar los asistentes de código.](#)

Generación de código



En el [siguiente video](#) te mostramos cómo insertar y generar código en Eclipse.

Resumen

Has finalizado esta lección. Veamos los puntos más importantes que hemos tratado.

En esta unidad hemos visto qué es un entorno de desarrollo y sus funciones básicas. Ha quedado constatada la importancia que tienen estas herramientas en la vida profesional de un desarrollador y lo importante que es tener un conocimiento de las mismas.

También te hemos mostrado una serie de vídeos para aprender a manejar con soltura Eclipse; recuerda que es uno de los IDE más utilizados.

Puedes ampliar la información:

Entornos de desarrollo

En este enlace puedes ampliar la información sobre los entornos de desarrollo.

[ENTORNOS DE DESAR...](#)

Manual de Eclipse

En este documento encontrarás una guía para novatos de Eclipse.

[MANUAL DE ECLIPSE](#)

Manual NetBeans

En este enlace puedes ver una guía rápida de NetBeans.

[MANUAL NETBEANS](#)



PROEDUCA