FP Informática y Comunicaciones

G.S. Desarrollo de Aplicaciones Multiplataforma | Desarrollo de Aplicaciones Web

Juan Carlos Moreno Pérez

ENTORNOS DE DESARROLLO

Segunda edición

EDITORIAL SINTESIS

Entornos de desarrollo

Juan Carlos Moreno Pérez

(segunda edición)

EDITORIAL SINTESIS

Vallehermoso, 34. 28015 Madrid
Teléfono 91 593 20 98
www.sintesis.com

ISBN: 978-84-1357-494-3
Depósito Legal: M-11.652-2026

Impreso en España - Printed in Spain

ÍNDICE

2. Evaluación de entornos integrados de desarrollo RA2

3. Diseño y realización de pruebas RA3

4. Optimización y documentación RA4

5. Elaboración de diagramas de clases RA5

6. Elaboración de diagramas de comportamiento RA6

2 Evaluación de entornos integrados de desarrollo

RESULTADO DE APRENDIZAJE Y CRITERIOS DE EVALUACIÓN

RA 2. Evalúa entornos integrados de desarrollo analizando sus características para editar código fuente y generar ejecutables.

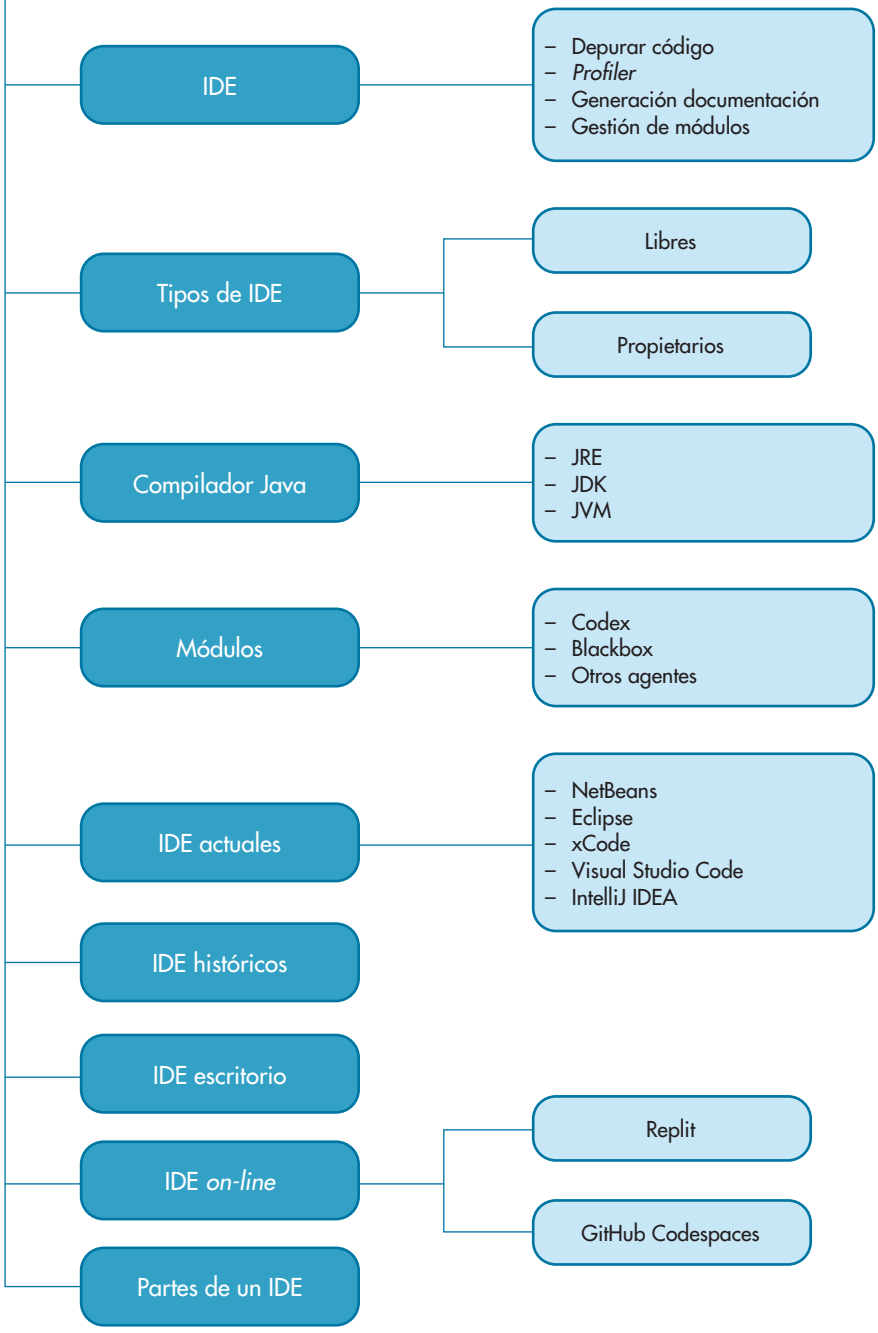
a) Se han instalado entornos de desarrollo, propietarios y libres.
b) Se han añadido y eliminado módulos en el entorno de desarrollo.
c) Se ha personalizado y automatizado el entorno de desarrollo.
d) Se ha configurado el sistema de actualización del entorno de desarrollo.
e) Se han generado ejecutables a partir de código fuente de diferentes lenguajes en un mismo entorno de desarrollo.
f) Se han generado ejecutables a partir de un mismo código fuente con varios entornos de desarrollo.
g) Se han identificado las características comunes y específicas de diversos entornos de desarrollo.

Objetivos de Desarrollo Sostenible

En este capítulo se va a trabajar el ODS 4.

MAPA CONCEPTUAL

EVALUACIÓN DE ENTORNOS INTEGRADOS DE DESARROLLO



GLOSARIO

CDDL. Sigla del inglés *common development and distribution license* (licencia común de desarrollo y distribución). De Sun Microsystems y basada en la MPL *(Mozilla public license)*.

EPL. Sigla del inglés *Eclipse public license*. Utilizada por la Fundación Eclipse para su software.

GPL. Sigla del inglés *general public license*. Es la licencia de software libre y código abierto más utilizada en la actualidad. Creada por Richard Stallman (el creador de la Free Software Foundation o FSF). Existen varias versiones de esta licencia las cuales van incorporando mejoras (hay que comprender que la versión 1 data de 1989).

Gradle. Herramienta de gestión y automatización de proyectos Java, Kotlin y Android.

IDE. Acrónimo del inglés *integrated development environment* (entorno de desarrollo integrado). Entorno donde el programador tiene todas las herramientas de trabajo a su disposición.

JDK. Sigla del inglés *Java development kit* (kit de desarrollo Java).

JRE. Sigla del inglés *Java runtime environment* (entorno de ejecución Java). Librerías básicas para ejecutar programas Java.

JVM. Sigla del inglés *Java virtual machine* (máquina virtual Java).

Licencia propietaria. Software que se distribuye en formato binario. No se ofrece acceso al código fuente. Generalmente, este software se vende con los derechos restringidos.

Maven. Herramienta de gestión y automatización fundamentalmente de proyectos Java.

Plugin. Complemento que se añade a otra herramienta para incrementar su funcionalidad.

PUNTO DE PARTIDA

Mathew trabaja en una empresa de sistemas de transporte inteligente en la cual se crean aplicaciones para que los autobuses, trenes y otros vehículos públicos de la empresa estatal se comuniquen con el centro de control y de esa manera se garantice tanto la eficiencia como el cumplimiento de los horarios.

Necesitan realizar una aplicación Java en los distintos vehículos que se comunique vía 5G con la central y Dimas, el encargado de Mathew, le ha propuesto que evalúe los diferentes entornos de desarrollo que puedan utilizar priorizando si es un entorno libre.

También le pide que evalúe la personalización y automatización del entorno, disponibilidad de módulos y *plugins* así como la posibilidad de integrar JUnit.

Por otro lado, le pide que, como síntesis, cree una tabla identificando las características comunes y específicas de los entornos de desarrollo elegidos para en una posterior reunión elegir la mejor opción.

Al preguntarle Matthew cuántos IDE tiene que evaluar, Dimas le responde que estudie en profundidad solamente tres.

2.1. Introducción

Los entornos de desarrollo son las herramientas con las cuales los programadores crean aplicaciones. Es cierto que pueden programarse con un editor y un compilador (a veces, con un depurador), pero, en entornos profesionales, casi siempre se utiliza un IDE. Un IDE consta de las siguientes herramientas:

1. *Editor.* Generalmente, se utilizan editores que colorean la sintaxis para ayudar al programador a comprender mejor el programa y detectar los errores más fácilmente.
2. *Compilador o intérprete.* Dependiendo del tipo de lenguaje utilizado, se necesitará para ejecución el intérprete o el compilador para generar código ejecutable.
3. *Depurador (intérprete).* Un buen depurador siempre tiene un intérprete detrás para ir ejecutando órdenes paso a paso, inspeccionar el valor de variables, etc.
4. *Constructor de interfaces gráficos.* Con él, el desarrollador podrá crear ventanas, botones, campos de texto, literales, pestañas, tablas, etc. Tiene todos los componentes que pueden encontrarse en una interfaz.
5. *Complementos y* plugins *(opcional).* Los complementos permiten extender las funcionalidades del IDE sin modificar sus partes principales. El objetivo de los mismos es poder cambiar la apariencia, añadir soporte para nuevos lenguajes, integrar herramientas como GIT o Bitbucket, asistentes de código, etc.

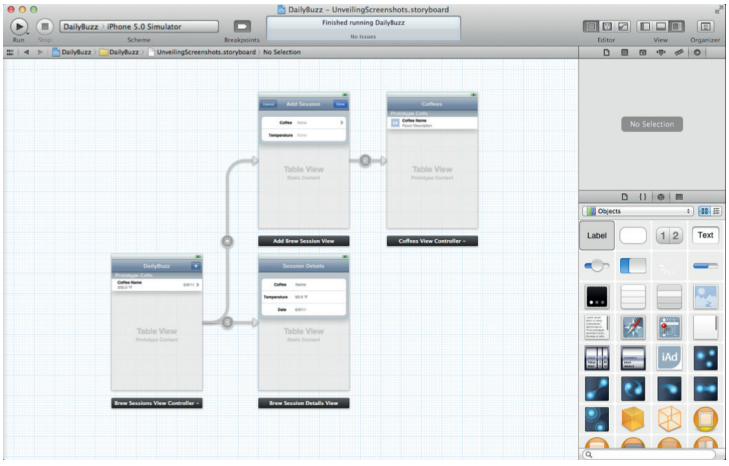


Figura 2.1
Xcode StoryBoard

2.2. Los primeros entornos de desarrollo

Una vez conocemos las herramientas para implementar entornos de desarrollo, veamos los más importantes.

2.2.1. Turbo Pascal

Lo lanzó la empresa Borland en el año 1983 y fue el IDE más potente de su época. Al principio, funcionaba en MS-DOS, CP/M y CP/M 86 y Macintosh, aunque posteriormente se creó una versión para Windows que tuvo mucho éxito.

Figura 2.2
Turbo Pascal versión 5.5 de Borland

Se lanzaron siete versiones y, en las últimas, podía utilizarse el ratón. Soportaba múltiples archivos en el mismo editor (diferentes ventanas) y podía programarse orientado a objetos. También poseía una herramienta llamada Turbo Profiler que permitía optimizar el código.

Fue una revolución en su época. La rapidez de compilación era asombrosa. De hecho, los compiladores actuales son más lentos.

Tras el éxito de esta herramienta, Borland creó nuevas herramientas, como Delphi, basadas en el mismo lenguaje de programación: Pascal.

ACTIVIDAD PROPUESTA 2.1

Indica si las siguientes afirmaciones son verdaderas o falsas y razona tus respuestas:

V F Para ejecutar un programa Java, es necesario tener el JDK.

V F Un programa con licencia propietaria se distribuye en formato binario.

Verificar

2.2.2. Visual Basic 6

Visual Basic 6 fue uno de los IDE más utilizados en su época, si no el que más. Este nuevo tipo de herramientas creó el paradigma de desarrollo RAD, acrónimo del inglés *rapid application development* (desarrollo rápido de aplicaciones). Un paradigma en el que primero se desarrollaban de una manera rápida las interfaces y se consensuaban con el usuario. Cuando se tenía el visto bueno, empezaban a crearse la base de datos y el código. Fue un cambio en el modelo de programar.

Los programadores creaban las interfaces a partir de una serie de componentes que ofrecía la propia herramienta. También podían utilizarse componentes de terceros, con lo cual se ganaba en funcionalidad y potencia.

El acceso a las bases de datos se realizaba utilizando DAO, RDO o ActiveX Data Objects, este último más rápido y más optimizado.

Visual Basic se utiliza en la actualidad gracias a que las macros realizadas en Office utilizan un dialecto suyo: *Visual Basic for applications* (VBA). Las macros son una herramienta muy potente, dado que combinan las características de Office con la potencia de todo un lenguaje de programación orientado a objetos.

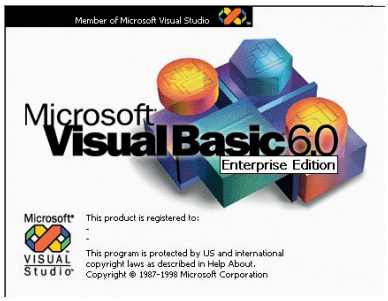


Figura 2.3
Splash de carga del Visual Basic 6

2.2.3. Delphi

Turbo Pascal fue un líder en su época y otro grande de la informática (Microsoft) sacó al mercado Visual Basic. Visual Basic era un IDE para Windows que hizo que Borland sacara algo más tarde al mercado Delphi, que fue una evolución del Turbo Pascal hacia el sistema Windows igual que Builder C++ fue la evolución del Turbo C.

Además de Delphi, Borland también saco al mercado el JBuilder. Un IDE de Java que tenía la ventaja de estar disponible también en Linux.

Delphi también tuvo su hermano de Linux llamado *Kylix,* que, desafortunadamente, se abandonó tras la versión 3.0, pero tenía la ventaja de que cualquier proyecto realizado en Windows podía recompilarse en Linux, y viceversa (siempre que se utilizasen los controles estándar).

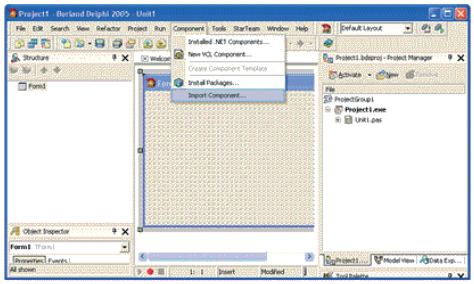


Figura 2.4
Delphi IDE

2.2.4. Visual C++

Visual C++ es un IDE para programar en C y C++. Su potencia radica en que incluye las bibliotecas de Windows, las *Microsoft Foundation classes* (MFC) y el *framework* .NET.

Es un IDE pesado, pero a la vez potente, puesto que, además de las bibliotecas propias, pueden añadírsele otras nuevas como DirectX, wxWidgets o SDL.

Al igual que Java, .NET ha incluido una herramienta bastante útil para autogestionar la memoria: el recolector de basura o *garbage collector*.

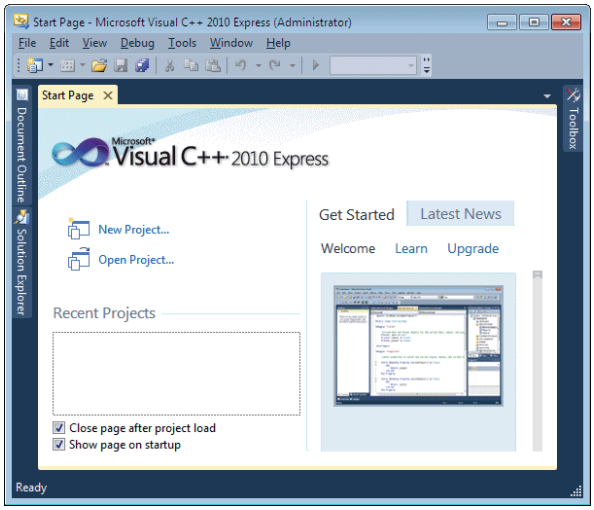


Figura 2.5
Visual C++ IDE

2.3. Entornos de desarrollo actuales

2.3.1. Xcode

Figura 2.6
Logo de Xcode

Xcode es la herramienta para realizar aplicaciones (*app*) para dispositivos Apple. Con esta herramienta, podrán realizarse aplicaciones nativas para iOS y OS X.

Si desea descargarse una versión antes de que se encuentre disponible para todo el mundo, hay que hacerse desarrollador de Apple.

Actualmente, no cuesta nada darse de alta como desarrollador, es gratuito, lo que cuesta es subir una aplicación a la App Store (la suscripción es de unos 100 dólares al año y pueden subirse todas las aplicaciones que se desee).

Con las nuevas versiones, ya puede programarse en Swift, mientras que, con las versiones anteriores, solamente puede programarse con Objetive C. Objetive C es un lenguaje parecido a Java/C/C++, pero con una sintaxis algo diferente. Muy potente y orientado a objetos.

2.3.2. Visual Studio Code

Figura 2.7
Logo Visual Studio Code

VS Code es un editor multiplataforma (Linux, Windows y MacOS) que destaca por ser rápido y ligero. El *marketplace* para instalar extensiones es muy amplio y permite adaptarlo a cualquier *framework*.

Tiene integración con Git y permite colaboración en tiempo real (*pair programming*) gracias a Live Share.

Es multilenguaje puesto que puede utilizarse para programar en Java, Python, JavaScript, HTML, CSS, C++, etc.

Muchos programadores junior y con más experiencia lo prefieren frente a otros gracias a su IntelliSense, el autocompletado inteligente de código que permite programar mucho más rápido.

2.3.3. IntelliJ IDEA

Figura 2.8
Logo Intellij IDEA

Este IDE es muy apreciado por los programadores de Java y Kotlin puesto que su autocompletado es capaz de comprender el código y ofrecer sugerencias precisas mediante sus algoritmos de *machine learning* (aprendizaje automático).

Tiene un analizador de código en tiempo real capaz de detectar errores y sugerir mejoras en el código al igual que permite una refactorización avanzada que en Java es capaz de extraer métodos o mover clases con seguridad.

Al igual que otros IDE, tiene soporte nativo para Git, Maven y Gradle sin tener que instalar ninguna extensión.

Muchos programadores del *framework* Spring Boot utilizan este IDE dado su soporte de primer nivel y su integración con JUnit o TestNG para la realización de pruebas unitarias.

2.3.4. Eclipse

Es un IDE de código abierto, gratuito y multiplataforma (Windows, Linux y MacOS). Al contrario que otros clientes livianos, es una plataforma potente con un buen editor, depurador y compilador. El JDT *(Java development toolkit)* es de los mejores que existen en el mercado y tiene detrás una gran comunidad de usuarios que van añadiendo mejoras al software. Permite gestionar varios proyectos a la vez gracias a los *workspaces* (espacios de trabajo). Ofrece integración nativa con Git y herramientas de automatización como Maven y Gradle.

Figura 2.9
Logo Eclipse IDE

2.4. Entornos de desarrollo *online*

Los entornos de desarrollo online o en la nube están extendiéndose cada vez más. Pese a la desventaja de la potencia, poseen muchas otras ventajas como el trabajo colaborativo, los repositorios comunes, el poder trabajar con cualquier dispositivo, etc.

Estas ventajas hacen que muchos desarrolladores y empresas de desarrollo opten por entornos en la nube.

Actualmente el IDE online más utilizado es Replit con la ventaja que soporta más de 50 lenguajes y los integrantes de un proyecto pueden colaborar en tiempo real. No hay que dejar de lado a GitHub Codespaces el cual tiene la ventaja de integrarse completamente con GitHub y es muy utilizado por profesionales y en proyectos Open Source.

RECURSO WEB

Hoy en día es muy fácil crear una aplicación con un agente en Replit. Visualiza el siguiente vídeo y aprende cómo los agentes IA te ayudan a programar aplicaciones de todo tipo.

ACTIVIDAD PROPUESTA 2.2

Indica si las siguientes afirmaciones son verdaderas o falsas y razona tus respuestas:

V F Desarrollar aplicaciones para iOS es gratuito, pero no subirlas a la App Store.

V F Con Intellij IDEA puede programarse en JavaSript.

Verificar

ACTIVIDAD GRUPAL 2.1

El ODS 4: Educación de calidad persigue garantizar una educación inclusiva, equitativa y de calidad, promoviendo competencias digitales y el aprendizaje permanente. Investiga en grupo sobre los agentes CLI e intenta responder a las siguientes preguntas:

- ¿Qué es un agente CLI y en qué te puede ayudar en un IDE?
- ¿Qué agentes CLI puedo instalar en Intellij IDEA?
- ¿Qué puede hacer JetBrains en Intellij IDEA?

¿Cómo puedo utilizar un agente CLI como Codex para aprender a programar? Intenta explicar cómo puede ofrecerte ejercicios, revisarlos y darte pistas para poder llegar tú al resultado correcto.

2.5. Entornos de desarrollo libres y propietarios

Existen muchos IDE Y, dependiendo de la popularidad del lenguaje, habrá más o menos opciones. En el cuadro 2.1, se ofrece una lista de IDE para los lenguajes Java y JavaScript.

CUADRO 2.1. IDE principal para los lenguajes Java y JavaScript

	IDE	Licencia	Windows	Linux	Mac OS X
Lenguaje Java	IntelliJ IDEA	Freemium (Community/Ultimate)	Sí	Sí	Sí
	Eclipse	Gratis (EPL)	Sí	Sí	Sí
	NetBeans	Gratis (Apache 2.0)	Sí	Sí	Sí
	Android Studio	Apache (Gratis)	Sí	Sí	Sí
	JDeveloper	Propietaria	Sí	Sí	Sí
Lenguaje JavaScript	VS Code	Código Abierto (MIT)	Sí	Sí	Sí
	WebStorm	Pago (Propietaria)	Sí	Sí	Sí
	Sublime Text	Evaluación	Sí	Sí	Sí
	Cursor	Freemium	Sí	Sí	Sí
	Atom (Community)	MIT (Gratis)	Sí	Sí	Sí

2.6. Instalación de un entorno integrado de desarrollo

La instalación de un entorno integrado de desarrollo cada vez es más sencilla e intuitiva. Prácticamente la mayoría de IDE permite identificar el lenguaje de programación y descargarse el complemento para compilar o interpretar el código fuente si no lo tienen ya instalado. A continuación, se explica cómo funciona el núcleo del compilador de Java, el JDK y el JRE.

2.6.1. El compilador de Java

El compilador de Java, también llamado *javac,* se encapsula dentro de un paquete de desarrollo que se llama *JDK,* del inglés *Java development kit* (equipo de desarrollo Java).

Para programar en Java, se necesita el compilador y, por lo tanto, habrá que instalar un JDK en la máquina donde vaya a desarrollarse.

Para ejecutar los programas desarrollados en Java, el sistema donde se ejecute deberá tener un JRE, del inglés *Java runtime environment* (entorno de ejecución Java), el cual contendrá una JVM, del inglés *Java virtual machine* (máquina virtual Java).

Java es multiplataforma, por lo tanto, no hay que compilar cada programa para cada sistema operativo, ya que, cuando se compila un programa, funcionará en cualquier sistema siempre y cuando tenga instalada la JVM correspondiente.

Téngase en cuenta que cada sistema operativo tendrá una JVM diferente.

RECUERDA

- JDK es el *Java development kit.* Es el software utilizado por los desarrolladores. Incluye el compilador de Java (javac), JRE y JVM.
- JRE es el *Java runtime environment.* Es el software utilizado por los usuarios. Este software incluye la JVM.
- JVM o *Java virtual machine.* Es el programa que ejecuta el código Java previamente compilado con el compilador de Java (javac).

2.6.2. Dudas frecuentes sobre el compilador de Java

A) Cómo sé si ya está instalado el JVM

Bastará con ejecutar el siguiente 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)
```

B) Cómo sé si ya está instalado el JDK

Bastará con ejecutar el siguiente comando:

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

C) Qué hay que hacer para instalar el JRE y el JDK

En Ubuntu, existe una versión de JRE y JDK en los repositorios. Instalarla es sumamente fácil.

A continuación, se muestran los comandos para su instalación:

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

Los dos primeros comandos son para actualizar el sistema y los paquetes en el caso de que no lo estén.

El tercero es para instalar el JRE y el cuarto (para el desarrollador) para instalar el JDK.

ACTIVIDAD PROPUESTA 2.3

¿Qué es JSX? ¿Existen entornos de desarrollo que trabajen con este lenguaje?

2.7. Depurar un programa

Es el momento de aprender a depurar un programa. Ningún programa suele funcionar a la primera ni será tal y como se diseñó en un primer momento. Siempre hay que depurar algunos fallos o simplemente verificar que lo que está haciendo lo hace de forma correcta.

En este caso, se muestra cómo hacerlo con NetBeans. A continuación, va a utilizarse el depurador para establecer un punto de ruptura y analizar el valor de las variables con las que está trabajándose.

Depurar un programa con NetBeans

Si, por el contrario, se está utilizando otro IDE como Visual Studio Code, el proceso es muy similar. Se colocan *breakpoints* (puntos de interrupción) en las líneas de código donde se desea detener y se inicia la depuración (F5 o botón de *debug*).

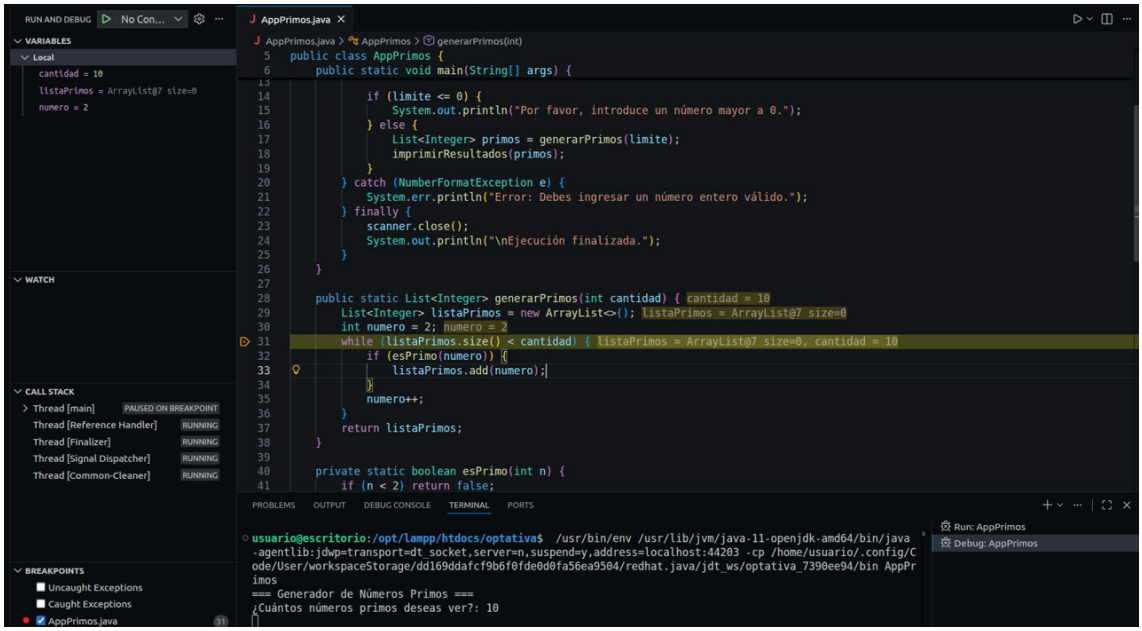


Figura 2.10
Debug de un programa Java en Visual Studio Code

Como se puede observar en la imagen anterior, en la barra de depuración de la izquierda aparecen las variables locales (también en el editor) con sus respectivos valores actuales y se pueden añadir expresiones *(watches)* para analizar el código de una forma más exhaustiva.

El depurador como en cualquier IDE permite todo tipo de acciones en tiempo real como habilitar o deshabilitar *breakpoints*, ejecutar línea a línea o hasta el siguiente *breakpoint,* etc. La barra de herramientas de depuración de VS Code es muy útil y intuitiva para depurar un programa Java.

Figura 2.11
Barra de herramientas de depuración en Visual Studio Code

2.8. Profiler. Análisis de aplicaciones

Muchas veces, cuando una aplicación está completamente desarrollada o en periodo de pruebas, es preciso analizar su rendimiento. NetBeans proporciona una herramienta para monitorizar los hilos de ejecución, el rendimiento de la CPU, el uso de memoria, etc.

Analizando el sistema por primera vez

Análisis de la memoria

ACTIVIDAD PROPUESTA 2.4

Busca información sobre el producto IntelliJ IDEA. Prueba a ver si puedes instalarlo en tu equipo. Muchas veces, los productos comerciales tienen ediciones reducidas que pueden obtenerse gratuitamente por tiempo limitado.

Compara este IDE con NetBeans o Eclipse.

Visual Studio Code también tiene herramientas para "perfilar" o, mejor dicho, monitorizar una aplicación. Con la extensión JProfiler, al lanzar una aplicación se pueden identificar cuellos de botella al analizar la CPU, memoria o subprocesos y también permite funcionar directamente con tareas de Gradle lo cual es ideal para aplicaciones Java SE/EE (Standard Edition o Enterprise Edition).

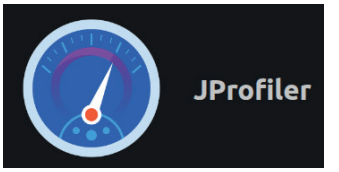


Figura 2.12
Plugin JProfiler para Visual Studio Code

2.9. Generación automática de documentación

Las aplicaciones o programas tienen que estar perfectamente documentados, pues, de lo contrario, sería muy difícil mantener el código. En Java, la documentación del código se escribe dentro del propio lenguaje, lo cual es verdaderamente útil. Java, además, tiene una herramienta que se llama *Javadoc* que extrae los textos y comentarios del código fuente y los transforma en páginas web (formato HTML).

En el capítulo 4 de este libro, se explicará en profundidad esta herramienta.

En Visual Studio Code y otros IDE, se pueden instalar *plugins* que analizan el código y automáticamente generan comentarios con la documentación técnica. En la siguiente imagen se puede observar que, al ejecutar el comando, se genera entre otra información, los parámetros de entrada y salida en los métodos de clase.

```java
/**
 * @param cantidad
 * @return List<Integer>
 */
public static List<Integer> generarPrimos(int cantidad) {
    List<Integer> listaPrimos = new ArrayList<>();
    int numero = 2;
    while (listaPrimos.size() < cantidad) {
        if (esPrimo(numero)) {
            listaPrimos.add(numero);
        }
        numero++;
    }
    return listaPrimos;
}
```

Figura 2.13
Resultado de ejecutar generar documentación en el IDE

2.10. Gestión de módulos

Los entornos como NetBeans aumentan su potencia gracias a la gestión de módulos o *plugins*. Con estos módulos, pueden crearse informes, trabajar con otros lenguajes de programación que no sean Java, etc.

Cómo añadir un módulo a NetBeans

Crear un nuevo proyecto Python en NetBeans

Cómo eliminar un plugin

RECURSO WEB

Accede a la página web de *plugins* de NetBeans, donde encontrarás cientos de plugins perfectamente clasificados y ordenados.

Para Visual Studio Code existen numerosos *plugins* o módulos. Los más utilizados por los programadores son los siguientes:

Plugin	Características
Python	Es un básico para los programadores de Python puesto que les permite utilizar IntelliSense, depuración y soporte para *notebooks* de Jupyter (aplicación web que permite ver fragmentos de código mientras que escribes. Muy utilizado en Python).
Github Copilot	Permite programar de forma más rápida gracias a la IA. Aunque es de pago es de los *plugins* más usados.
Prettier	Permite mantener el código más organizado y ordenado. Se usa sobretodo en JavaScript, TypeScript, HTML y CSS.
Docker	Permite gestionar de una forma sencilla los contenedores (paquetes de código que se pueden ejecutar de manera fiable en cualquier máquina) desde el mismo editor.
GitLens	Mejora todas las funciones de Git y permite comprender, escribir y revisar mejor el código del repositorio.

ACTIVIDAD PROPUESTA 2.5

Indica si las siguientes afirmaciones son verdaderas o falsas y razona tus respuestas:

V F Mediante una herramienta llamada *Javadoc,* es posible analizar el uso de la memoria de un programa Java.

V F Aunque puede programarse dn Java con NetBeans, no es posible programar en JavaScript.

Verificar

ACTIVIDAD GRUPAL 2.2

4 EDUCACIÓN DE CALIDAD

El ODS 4: Educación de calidad promueve metodologías que mejoren la calidad del aprendizaje y desarrollen competencias relevantes para el ámbito profesional. Investiga en grupo en qué consiste el *pair programming* e intenta responder a las siguientes preguntas junto con tu compañero:

- ¿En qué consiste el *pair programming*?
- ¿Cómo las nuevas herramientas pueden ayudarte a aprender *debugging*?
- ¿Cómo se pueden aprender buenas prácticas con un agente IA?
- ¿Existe alguna herramienta que te ayude a realizar un proyecto desde cero explicándote paso a paso cómo lo desarrolla? Explica cómo.
- ¿Crees que *pair programming* puede ser un salto en la calidad del aprendizaje?