Entornos de desarrollo

Entornos de desarrollo

Juan Carlos Moreno Pérez

EDITORIAL SINTESIS

Asesor editorial: Juan Carlos Moreno Pérez
© EDITORIAL SÍNTESIS, S. A.
Vallehermoso, 34. 28015 Madrid
Teléfono: 91 593 20 98
www.sintesis.com

ISBN: 978-84-9171-161-2
Depósito Legal: M-22.893-2018

Impreso en España - Printed in Spain

Índice

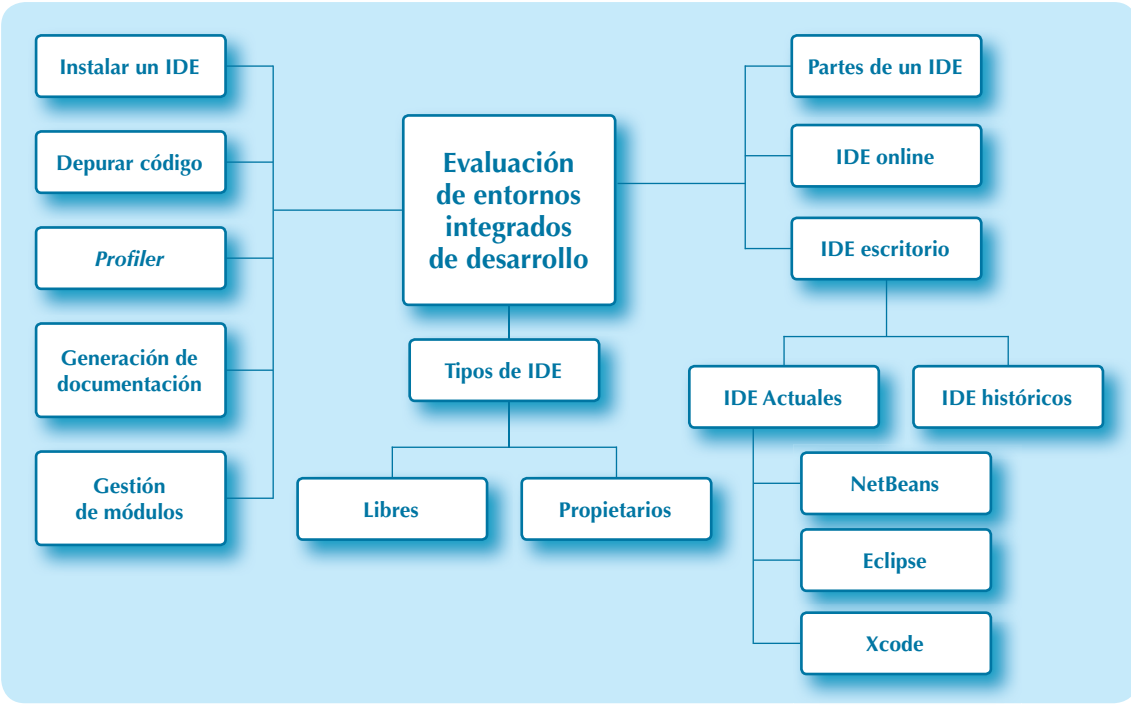
2

Evaluación de entornos integrados de desarrollo

Objetivos

- ✓ Con este capítulo, vas a aprender cómo instalar un IDE y cómo configurarlo para sacarle todo el rendimiento que este ofrece.
- ✓ Existen muchos tipos de IDE, desde los entornos de escritorio a los entornos en la nube pasando por los entornos de desarrollo libre y propietarios. Los IDE van evolucionando y cambiando, por lo que el programador tendrá que ir adaptándose a ellos. Por ello, también se hace un repaso a los IDE más exitosos de los últimos tiempos.
- ✓ Otra de las destrezas que tienes que aprender es depurar cualquier código. Es básico saber depurar un código paso a paso desde el lugar que sea preciso y saber utilizar adecuadamente los puntos de ruptura o *breakpoints*.
- ✓ Por último, también es importante que sepas cómo generar documentación de forma automática y conozcas cómo ampliar las capacidades del IDE mediante *plugins*.

Mapa conceptual



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).

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.

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

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.

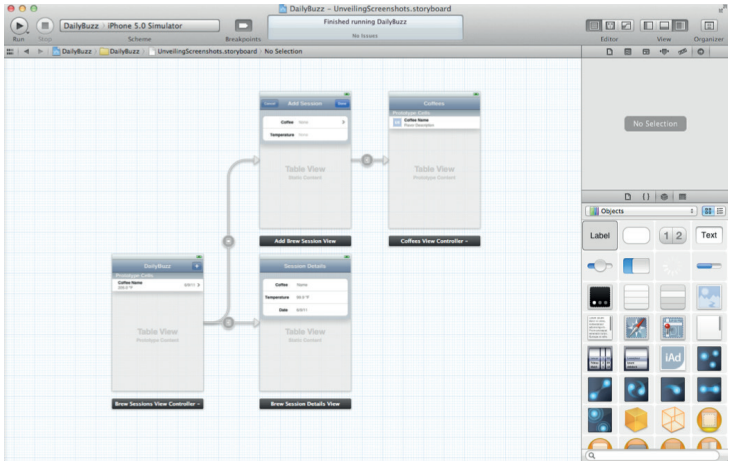


Figura 2.1
Xcode StoryBoard.

2.2. Los primeros entornos de desarrollo

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.

Actividades propuestas

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

V F 2.1. Para ejecutar un programa Java, es necesario tener el JDK.

V F 2.2. 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.

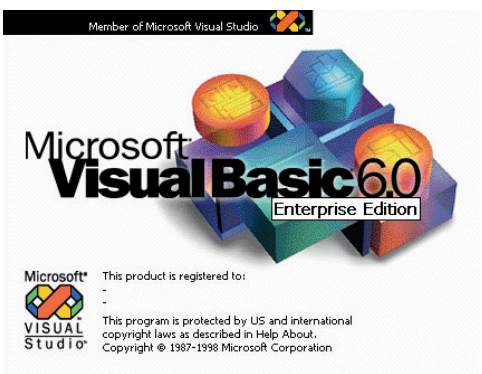


Figura 2.3
Splash de carga del Visual Basic 6.

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.

2.2.3. Delphi

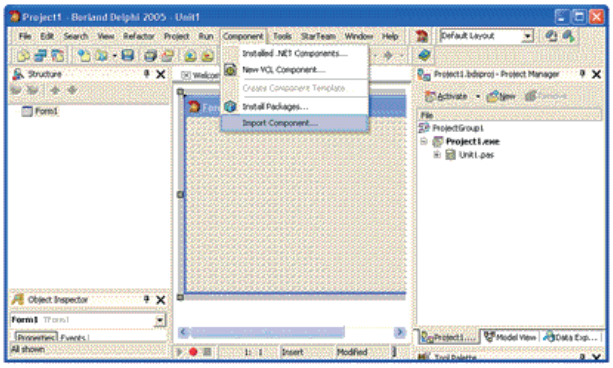


Figura 2.4
Delphi IDE.

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).

2.2.4. Visual C++

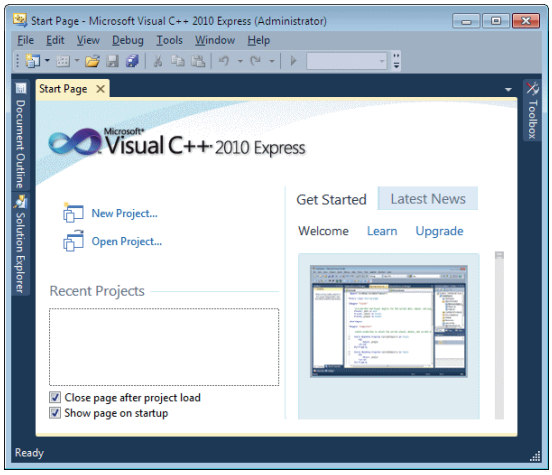


Figura 2.5
Visual C++ IDE.

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.*

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. NetBeans

Figura 2.7
Logo NetBeans.

NetBeans está escrita en Java, lo que la convierte en una plataforma disponible para un gran número de sistemas operativos (Windows, Linux o Mac OS X). Se creó para desarrollar aplicaciones en Java, pero también puede programarse con ella en Python, PHP, HTML5 y C/C++.

Es *open source* lo que hace que muchos programadores se decanten por este IDE. De hecho, cuando sale una nueva versión al mercado, suele estar bastante probada.

Se basa en la modularidad. Todas las funciones las realizan módulos, los cuales pueden ir añadiéndose según necesidades del programador. De hecho, cuando se descarga, tiene todos los módulos de Java incluidos por defecto. Muchas herramientas están basadas en NetBeans como Sun Studio, Sun Java Studio Creator y otras más.

Contiene una herramienta para crear interfaces de usuario (llamada al comienzo *Matisse*). Esta herramienta permite crear aplicaciones basadas en la librería Swing.

En el editor, puede programarse también en JavaScript, Ajax y CSS.

2.3.3. Eclipse

Es un IDE de código abierto. Al contrario que otros clientes livianos, es una plataforma potente con un buen editor, depurador y compilador (el ECJ). 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.

Fue desarrollado por IBM como evolución de su VisualAge, pero ahora lo mantiene la fundación Eclipse, que es independiente y sin ánimo de lucro.

Tenía licencia CPL *(common public license)*, pero luego la fundación cambió dicha licencia por una EPL *(Eclipse public license)*.

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.

Veamos cómo funcionan dos IDE en la nube:

Figura 2.8
Logo Eclipse IDE.

Cloud 9

CodeAnywhere

Recurso web

Accede a MyFPschool, donde puede verse en un vídeo cómo instalar en Linux un IDE en la nube como es CodeBox.

Actividades propuestas

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

V F 2.3. Desarrollar aplicaciones para iOS es gratuito, pero no subirlas a la App Store.

V F 2.4. Con NetBeans, puede programarse en JavaScript.

Verificar

2.5. Entornos de desarrollo libres y propietarios

Existen muchos IDEY, 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 para los lenguajes Java y JavaScript

	IDE	Licencia	Windows	Linux	Mac OS X
Lenguaje Java	Eclipse	EPL	Sí	Sí	Sí
	NetBeans	CDDL/GPL2	Sí	Sí	Sí

[.../...]

CUADRO 2.1 (CONT.)

	Visual Studio	Propietario	Sí	No	No
	JDeveloper	Propietario	Sí	Sí	Sí
Lenguaje JavaScript	Eclipse	EPL	Sí	Sí	Sí
	NetBeans	CDDL/GPL2	Sí	Sí	Sí
	Geany	GPL	Sí	Sí	Sí
	KDevelop	GPL	No	Sí	No
	JBuilder	Propietario	Sí	Sí	Sí
	JCreator	Propietario	Sí	No	No
	JDeveloper	Propietario	Sí	Sí	Sí

2.6. Instalación de un entorno integrado de desarrollo

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.

Investiga

¿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

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

Investiga

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.

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.

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

www

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

Actividades propuestas

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

V F 2.5. Mediante una herramienta llamada *Javadoc*, es posible analizar el uso de la memoria de un programa Java.

V F 2.6. Aunque puede programarse en Java con NetBeans, no es posible programar en JavaScript.

Verificar

¿Qué has aprendido?

- ✓ Durante este capítulo, se ha detallado cómo funciona un IDE, las posibilidades que ofrece y cómo ampliar su funcionalidad.
- ✓ Es importante que el desarrollador maneje de forma fluida el IDE porque los entornos actuales permiten automatizar muchas tareas, de tal manera que ahorran tiempo y mejoran la fiabilidad de los programas.
- ✓ La supremacía de los IDE de escritorio, últimamente, está perdiendo protagonismo frente a los IDE en la nube. La ventaja de estos últimos es que favorecen el desarrollo colaborativo y puedes trabajar con ellos en cualquier sitio que ofrezca una conexión a internet y con cualquier dispositivo ya sea tableta, portátil, ordenador o incluso *smartphone* (este último solo para realizar pequeñísimos cambios).

Resumen

- Un IDE puede constar de las siguientes herramientas:
  - Editor.
  - Compilador o intérprete.
  - Depurador (intérprete).
  - Constructor de interfaces gráficos.
- 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, poder trabajar con cualquier dispositivo, etc.
- Algunos entornos online son:
  - CodeBox.
  - Cloud 9.
  - CodeAnywhere.
- Existen entornos de desarrollo libres y propietarios.
- Para programar en Java, se necesitan los siguientes elementos:
  - 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).
- Para depurar un programa, hay que hacer uso de:
  - *Breakpoint*.
  - *Watch*.
- En algunos IDE, existe una herramienta que es el Profiler para monitorizar las aplicaciones.
- Javadoc es una herramienta para la generación automática de documentación en Java.
- En algunos IDE, es posible añadirle módulos o *plugins* para poder realizar proyectos en múltiples lenguajes de programación.

Ejercicios prácticos

1. Instalación de CodeBox.

Ten en cuenta lo siguiente:

- La página oficial de CodeBox en GitHub es la siguiente: https://github.com/CodeboxIDE/codebox.
- Tienes disponibles dos tipos de instalaciones: utilizando npm y con un instalador.