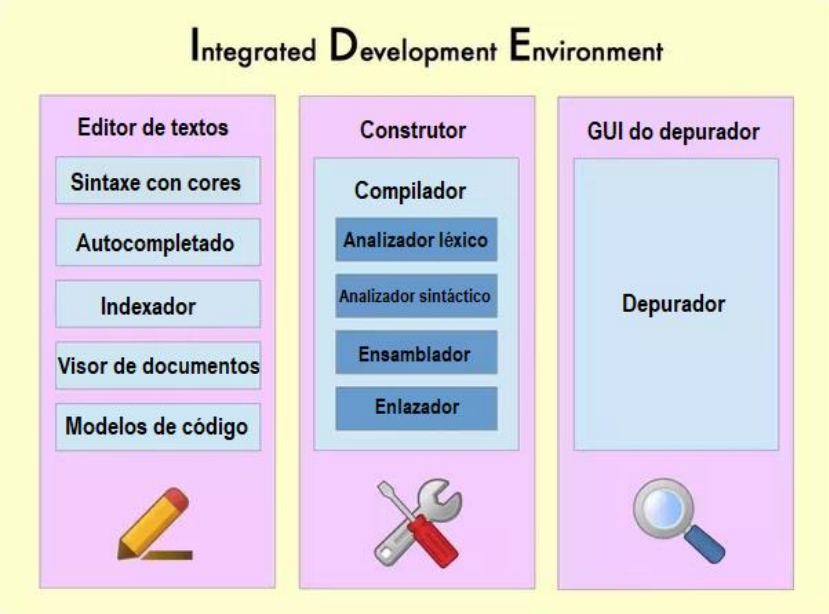
Contornos de desenvolvemento

Os contornos de desenvolvemento son as ferramentas coas que os programadores crean aplicacións. Aínda que pode programarse cun editor e un compilador, nos contornos profesionais soe utilizarse un contorno de desenvolvemento xa que axudan a aumentar a **produtividade** dos programadores.

Aos contornos de desenvolvemento chámaselles con frecuencia **IDE** (Integrated Development Environment – Contorno de Desenvolvemento Integrado) e apórtannos unha serie de ferramentas para que nos resulte máis doado programar e, normalmente, contan cun resaltado de sintaxe por defecto.



Un IDE é un contorno de programación empaquetado como un programa de aplicación e que, normalmente, consiste en:

- Un **editor** de texto para escribir código, con corrección automática de erros sintácticos.
- Un **compilador** para **traducir** o código escrito nunha **linguaxe de programación** a outro que sexa capaz de interpretar o aparato onde se executa.
- Un **depurador** para corrixir erros mentres o programa se executa.
- Un construtor de **interface gráfica** para **deseñar** as **ventás** que verá o usuario.
- Un **controlador de versións** para manter un rexistro de todo canto se ten feito.

O IDE agrupa as ferramentas indicadas normalmente nun contorno visual, de forma que o programador non necesite introducir múltiples comandos para compilar, interpretar, depurar, etc. Habitualmente teñen unha avanzada interface gráfica de usuario (GUI). Tamén soe existir a posibilidade de engadir **plugins** para incrementar as funcionalidades do IDE. Moitos IDEs permiten facer **probas unitarias** par que o código se probe automaticamente.

Os IDE ofrecen un marco de traballo **amigable** para a maioría das linguaxes de programación como poden ser Java, C++, C#, Python, Delphi, etc.

Un mesmo IDE pode traballar con **varias linguaxes** de programación.

Os primeiros contornos de desenvolvemento

Turbo Pascal. Foi lanzado pola empresa Borland en 1983 e foi o IDE máis potente da súa época. Durante a década dos 80 e ata principios dos 90 foi moi popular debido ás súas magníficas extensións e aos tempos de compilación sumamente curtos. Foi unha revolución para a súa época. A rapidez de compilación era asombrosa. De feito, os compiladores actuais son máis lentos.

Lanzáronse sete versións e, na última, podía usarse o rato. Na versión inicial comprimía nuns 50KB un editor, un compilador e un enlazador para crear arquivos executables.

Tras o éxito desta ferramenta, Borland creou novas ferramentas como Delphi, baseada na mesma linguaxe de programación, Pascal.

Microsoft Visual Basic 6. Foi un dos IDE máis utilizados na súa época. Este novo tipo de ferramenta creou o paradigma de desenvolvemento RAD (Rapid Application Development – Desenvolvemento Rápido de Aplicacións).

No paradigma RAD desenvolvíanse primeiro de maneira rápida as interfaces que se consensuaban co usuario. Cando os desenvolvedores tiñan o visto bo das interfaces, realizaban a programación. As interfaces creábanse a partir dunha serie de compoñentes que proporcionaba a propia ferramenta. VB6 só funcionaba no sistema operativo Windows.

Contornos de desenvolvemento actuais

NetBeans. NetBeans é un IDE escrito en Java. É un contorno moi utilizado polos programadores. Trátase dun contorno **multilinguaxe** e **multiplataforma** no que poder desenvolver software de calidade. Permite desenvolver aplicacións de escritorio, aplicacións móbiles e web. É un IDE libre e gratuíto, sen restricións de uso.

NetBeans é un dos IDE máis famosos para programar en Java.

NetBeans comezou como proxecto estudantil de estudantes universitarios de matemáticas e física en 1996 na república Checa. Sun MicroSystems adquiriu NetBeans no ano 2000. Posteriormente, NetBeans pasou a ser da compañía Oracle cando Oracle comprou en 2010 Sun MicroSystems. No ano 2016 Oracle enviou á incubadora de proxectos da organización sen fins de lucro Apache Software Foundation (ASF) o proxecto NetBeans.

Desenvolvéronse varias versións de NetBeans desde que Oracle donou NetBeans á ASF. A versión 11.0 de Apache NetBeans foi lanzada a principios

do mes de abril de 2019. A versión 12.0 de Apache NetBeans foi lanzada a principios do mes de xuño de 2020. O lanzamento de Apache Netbeans 12.0 caracterízase principalmente por chegar acompañado dun ciclo de soporte a longo prazo (**LTS** – Long Term Support). Isto significa que terá soporte e será actualizada durante máis tempo que unha versión normal. As versións LTS soen ser versións máis estables e probadas que o resto.

A linguaxe que mellor soporta é **Java**, xa que inicialmente foi creado para ser o IDE de Java. Non obstante é multilinguaxe xa que soporta JavaScript, HTML5, PHP, C/C++, etc. Ten bastantes módulos (plugins) que permiten ampliar as súas funcionalidades.

É multiplataforma xa que está dispoñible para un grande número de sistemas operativos como Windows, Linux ou MAC OS X.

Eclipse. Eclipse é un dos contornos de desenvolvemento máis coñecidos e utilizados polos programadores xa que se trata dun contorno de desenvolvemento de código aberto e multiplataforma. Eclipse creouse a partir do Visual Age de IBM pero agora manteno a Fundación Eclipse, que é independente e sen ánimo de lucro. Está soportado por unha comunidade de usuarios o que fai que teña moitos módulos (plugins) que permiten que serva para case calquera linguaxe de programación. Neste aspecto, é un dos mellores. Utilízase para Java, C++, PHP, Perl e moitas linguaxes máis.

Hai moitos tutoriais na rede tanto de instalación como de utilización.

O seu punto débil é que está lonxe de ser un contorno no que sexa sinxelo iniciarse e, polo tanto, para usuarios pouco experimentados pode ser un pouco duro.

Visual Studio. Foi deseñado por Microsoft e trátase dun dos mellores contornos de programación que existen, sempre e cando se utilicen linguaxes de Microsoft. Microsoft ten creado o Visual Studio Community que é moi parecido ao Visual Studio de pagamento só que está soportado pola comunidade. Este contorno permite facer aplicación web e de escritorio e axuda moito ao programador.

O inconveniente que ten é que só é válido para linguaxes de Microsoft tales como Visual C++, Visual C#, ASP.NET e Visual Basic.NET. Actualmente téñense desenvolvido extensións para moitas outras.

Visual Studio permite aos desenvolvedores crear aplicacións, sitios e aplicacións web, así como servizos web en calquera contorno que soporte a plataforma .NET.

Visual Studio Code. Visual Studio Code é un editor de programación multiplataforma desenvolvido por Microsoft. É un proxecto de software libre que foi lanzado baixa a **licenza MIT** co seu código fonte publicado en GitHub no repositorio **VSCode** (https://github.com/microsoft/vscode). É **gratuíto** e de **código aberto** aínda que a descarga oficial está baixo **software privativo** e inclúe características personalizadas por Microsoft. A páxina oficial de Visual Studio Code é https://code.visualstudio.com/.

A primeira versión beta de Visual Studio Code publicouse en novembro de 2015 e a primeira versión estable de Visual Studio Code 1.0 publicouse en abril de 2016. Desde a súa aparición, Visual Studio Code ten mantido un ritmo de desenvolvemento moi rápido, e publícase unha nova versión a principios de cada mes (excepto en xaneiro). Ademais, case todos os meses publícanse versións secundarias que corrixen fallos de última hora.

Visual Studio Code integra soporte de depuración, soporte de control de versións Git, resaltado de sintaxe, finalización intelixente de código, refactorización de código e fragmentos.

Dispón de múltiples extensións para ampliar as funcionalidades de Visual Studio Code.

Instalación de contornos de desenvolvemento

O compilador de Java

O compilador de Java, tamén chamado **javac**, encapsúlase dentro dun paquete de desenvolvemento chamado **JDK** (Java SE Development Kit – Equipo de Desenvolvemento de Java).

Para programar en Java necesítase o compilador e, polo tanto, haberá que instalar o seu **JDK** na máquina onde vaia facerse o desenvolvemento.

Para executar os programas desenvolvidos en Java, o sistema onde se execute deberá ter un **JRE** (Java Runtime Environment – Contorno de Execución de Java) que conterá a **JVM** (Java Virtual Machine – Máquina Virtual de Java).

Java é multiplataforma, polo tanto, non hai que compilar cada programa para cada sistema operativo, xa que, cando se compila un programa, funcionará en calquera sistema operativo, sempre e cando teña instalada a JVM correspondente.

Para saber se temos instalada a JVM no noso sistema podemos usar o seguinte comando na liña de comandos:

```
C:\>java -version
java version “1.8.0_221”
Java(TM) SE Runtime Environment (build 1.8.0_221-b11)
Java HotSpot(TM) Client VM (build 25.221-b11, mixed mode, sharing)
```

Para saber se temos instalado o JDK podemos usar o seguinte comando:

```
C:\>javac -version
javac 1.8.0_91
```

Hai 3 edicións de Java:

- Java Standard Edition (**Java SE**)
- Java Enterprise Edition (**Java EE**)
- Java Micro Edition (**Java ME**).

Java SE é unha plataforma que permite a execución e despregamento de aplicacións Java en computadoras de **escritorio** e **servidores**. Os dous produtos de Oracle que implementan a plataforma Java SE son o **JDK** (Java SE Development Kit) e o **JRE** (Java Runtime Environment).

Java EE é unha plataforma que está construída sobre a plataforma JSE. Esta plataforma subministra APIs adicionais e o contorno para o desenvolvemento e execución de **aplicacións empresariais** (as que requiren operacións a grande escala, nunha topoloxía distribuída, escalable, confiable e segura).

Java ME é o **subconxunto mínimo** de tecnoloxías e especificacións de Java para poder executar aplicacións en **dispositivos pequenos**.

Instalación do JDK e JRE predeterminados en Linux

Para a instalación eliximos un **servidor Ubuntu 18.04**.

A opción máis sinxela para instalar Java en **Ubuntu** é utilizar a versión que forma parte do paquete Ubuntu. Por defecto, Ubuntu 18.04 inclúe **OpenJDK**, que é

unha variante de código aberto de JRE e JDK de Oracle (licenza GPL) e que é mais que suficiente para que funcione o 99% do software que require Java.

Para os programadores, o principal problema desta versión de Java é que **non** permite a creación de **aplicacións comerciais** aínda sendo un software de código aberto.

Para instalar o JRE executamos o seguinte comando:

```
sudo apt install default-jre
```

Con este comando instálase o Java Runtime Environment (JRE). Isto permite executar case todo o software de Java.

Tras a instalación debe verificarse que se instalou correctamente.

```
java -version
openjdk version "11.0.11" 2021-04-20
OpenJDK Runtime Environment (build 11.0.11+9-Ubuntu-Oubuntu2.18.04.1)
OpenJDK 64-Bit Server VM (build 11.0.11+9-Ubuntu-Oubuntu2.18.04.1,mixed
mode, sharing)
```

Para instalar o JDK, debe executarse o seguinte comando:

```
sudo apt install default-jdk
```

Tras a instalación debe verificarse que se instalou correctamente o compilador de Java.

```
javac -version
javac 11.0.11
```

A instalación do JDK instala tamén o JRE no caso de que non se instalara previamente.

No caso de que queiramos **desintalar** o **OpenJDK** incluíndo as súas dependencias e os ficheiros de configuración usariamos o seguinte comando no terminal:

```
sudo apt-get purge --auto-remove openjdk*
```

Instalación de Oracle JDK 17 en Linux

Oracle JDK 17 é un kit de desenvolvemento LTS de Java. JDK 17 terá soporte comercial ata setembro de 2029.

O acordo de concesión de licenzas para Java de Oracle só permite a instalación automática mediante os administradores de paquetes para a última versión de Java.

Na páxina de descargas de Oracle:

https://www.oracle.com/es/java/technologies/javase-downloads.html

podemos ver que a última versión de Java é **Java SE 17**.

Ubuntu controla o software que está dispoñible nos seus repositorios e que versións.

Cando aparece unha nova aplicación ou unha nova versión dunha aplicación, esta non se incorpora de forma inmediata. En Ubuntu comproban que a nova aplicación ou a nova versión é compatible co sistema, para desta maneira contribuír á estabilidade do mesmo. A incorporación pode tardar semanas, meses ou quizás nunca.

Os **repositorios PPA** (Personal Package Archive) permítennos instalar e probar as novas versións aínda que non fosen aprobadas por Ubuntu.

Para saber que versións podemos instalar utilizando repositorios PPA podemos consultar a páxina:

https://launchpad.net/~linuxuprising/+archive/ubuntu/java/+packages

Tras consultar esta páxina, vemos que unha versión que podemos instalar é a Java 17.

Para facer a instalación primeiro instalamos as dependencias necesarias para engadir un novo repositorio:

```
sudo apt install software-properties-common
```

Habilitamos o PPA Linux Uprising executando o seguinte comando:

```
sudo add-apt-repository ppa:linuxuprising/java
```

Unha vez temos engadido o repositorio, actualizamos a lista de paquetes e instalamos oracle-java17-installer da seguinte maneira:

```
sudo apt update
sudo apt install oracle-java17-installer
```

Haberá que aceptar a licencia de Oracle.

Tras a instalación, debe comprobarse a mesma con java -version e javac -version.

Se queremos **desinstalar** o **Oracle JDK**, primeiro debemos comprobar os paquetes que están instalados:

```
sudo dpkg –list | grep -i jdk
```

Posteriormente borramos Oracle Java 17 completamente:

```
sudo apt-get purge oracle-java17-installer
```

```
sudo apt-get autoremove
```

Podemos comprobar que os paquetes se borraron:

```
sudo dpkg –list |grep -i jdk
```

Instalación de Oracle JDK 17 en Windows

Para instalar o Oracle JDK 17 en Windows é necesario descargalo da web de Oracle:

https://www.oracle.com/java/technologies/javase-downloads.html

Eliximos o **instalador para Windows** e dámoslle á ligazón para descargar.

Para poder descargar é necesario aceptar as condición da licenza e, tras premer o botón de descarga é necesario utilizar as **credencias de Oracle** para que se inicie a descarga do produto. Procedemos á instalación unha vez temos descargado o executable. Comprobamos que foi correcta na liña de comandos.

Compilación en Java na liña de comandos

Unha vez sabemos que temos instalado tanto o JRE como o JDK no sistema xa podemos compilar e executar programas Java desde a liña de comandos.

Podemos crear un arquivo de nome **Saudo.java** co seguinte contido:

```
public class Saudo {
    public static void main(String args[]) {
        System.out.println("Java en Linux");
    }
}
```

O nome da clase sempre ten que ser igual ao nome do arquivo sen ter en conta a extensión.

Para compilar este arquivo en liña de comandos temos que facer:

```
javac Saudo.java
```

Tras compilalo, se non hai erros de compilación, no cartafol onde se creou Saudo.java debe aparecer o arquivo **Saudo.class**.

Para executar o arquivo Saudo.class facemos:

```
java Saudo
```

Como resultado debe verse na pantalla a mensaxe: Java en Linux.

Os **paquetes en Java** (packages) son a forma na que Java nos permite agrupar dunha maneira lóxica os compoñentes dunha aplicación que estean relacionados entre si. Deste modo, os paquetes en Java axudan a darlle unha boa organización a unha apliación.

Imos crear un cartafol para almacenar o seguinte proxecto ProxectoSaudo no que imos usar un paquete.

Para facelo escribimos:

```
mkdir ProxectoSaudo
```

Dentro deste cartafol imos crear outro de nome paquete.

```
cd ProxectoSaudo
mkdir paquete
```

Para editar un arquivo na liña de comandos podemos facelo con nano:

```
cd paquete
nano SaudoPaquete.java
```

O código Java de **SaudoPaquete.java** pode ser:

```
package paquete;
public class SaudoPaquete{
    public static void main(String[] args){
        System.out.println("Saudo en paquete");
    }
}
```

Para compilar esta clase que está dentro dun paquete debemos facelo desde o cartafol ProxectoSaudo.

```
javac paquete/SaudoPaquete.java
```

Para executar tras a compilación facemos:

```
java paquete.SaudoPaquete
```

O resultado da execución será:

```
Saudo en paquete
```

Un **arquivo coa extensión JAR** (abreviatura de Java Archive) soe utilizarse para almacenar programas e xogos Java nun só arquivo. Algúns conteñen arquivos que traballan como aplicacións independentes e outros conteñen bibliotecas de programas para que outros os usen.

Os arquivos JAR están comprimidos en ZIP e soen almacenar arquivos CLASS, un arquivo manifesto e recursos de aplicacións como imaxes, clips de son e certificados de seguridade. Como poden conter centos ou incluso miles de arquivos nun formato comprimido, son doados de compartir.

Imos facer un arquivo JAR a partir dun novo proxecto que imos almacenar en ProxectoAutor.

Creamos o proxecto:

```
mkdir ProxectoAutor
```

A continuación, creamos a clase Autor no proxecto.

```
cd ProxectoAutor

nano Autor.java
```

O contido da clase Autor podería ser a seguinte:

```
public class Autor{
    private String nome;
    //construtor por defecto
    public Autor(){
        this.nome="Anónimo";
    }
    //construtor con 1 parámetro
    public Autor(String nome){
        this.nome=nome;
    }
    //getter do atributo nome
    public String getNome(){
        return this.nome;
    }
    //setter do atributo nome
    public void setNome(String nome){
        this.nome=nome;
    }
}
```

Podemos crear unha clase ProbarAutor.java para probar a clase Autor.

```
public class ProbarAutor{
    public static void main(String[] args){
        //creamos un novo autor
        Autor autor1=new Autor();
        //visualizamos o nome do autor1
        System.out.println(autor1.getNome());
        //creamos un novo autor
        Autor autor2=new Autor("Cunqueiro");
        //visualizamos o nome do autor2
        System.out.println(autor2.getNome());
    }
}
```

Para poder executar a clase ProbarAutor necesitamos ter tamén a clase Autor.

Para compilar podemos facer:

```
javac Autor.java

javac ProbarAutor.java
```

Unha vez temos compilados os dous arquivos, podemos executar:

```
java ProbarAutor
```

O resultado da execución sería:

```
Anonimo
Cunqueiro
```

Se queremos ter un único ficheiro a partir do cal executar e ter todos os arquivos class xerados nese arquivo, podemos crear un arquivo jar.

Creamos un arquivo **manifest.mf** para indicar onde está o punto de entrada ao programa, é dicir, o método main. Podemos crear o arquivo manifest.mf co nano.

O contido do manifest.mf sería:

```
Main-Class: ProbarAutor
```

Creamos o arquivo autores.jar usando o manifest.mf e todos os arquivo class xerados:

```
jar cfm manifest.mf autores.jar *.class
```

Podemos ver o contido de autores.jar do seguinte modo:

```
jar tf autores.jar
```

Para executar o arquivo facemos:

```
java -jar autores.jar
```

O resultado da execución será:

```
Anonimo
Cunqueiro
```

Se queremos realizar un programa que colla os datos da **liña de comandos** coma en **Produto2.java**:

```
public class Produto2 {
    public static void main(String args[]) {
        if (args.length>0){
            int num=Integer.parseInt(args[0]);
            System.out.println("O dobre de "+num+" e: "+(num*2));
        }else{
            System.out.println("Programa executado");
        }
    }
}
```

Para compilar **Produto2.java** facemos:

```
javac Produto2.java
```

Se non hai ningún erro de compilación, executamos facendo:

```
java Produto2 12
```

O resultado desta execución será: O dobre de 12 e: 24.

Se queremos facer un programa interactivo como **Produto2Interactivo.java** facemos:

```
import java.util.Scanner;
public class Produto2Interactivo {
    public static void main(String args[]) {
        Scanner in=new Scanner(System.in);
        System.out.println("Introduce un numero:");
        int num=in.nextInt();
        System.out.println("O dobre de "+num+" e: "+(num*2));
    }
}
```

Para compilalo facemos:

```
javac Produto2Interactivo.java
```

Para executalo, tras compilalo sen erros facemos:

```
java Produto2Interactivo
```

Se, durante a execución, indicamos que o número é 114 o resultado será: O dobre de 114 e: 228.

Se non queremos indicar de cada vez os datos da execución do programa, sobre todo se estamos depurando e non damos atopado o erro, podemos crear un arquivo de texto co nome que queiramos como pode ser **datos.txt** para gardar eses datos de proba. Por exemplo, o noso arquivo datos.txt pode ter gardado o número **80**. Se o programa pide máis dun número, poden poñerse estes números separados por espazos no ficheiro datos.txt.

Para traballar co arquivo datos.txt indicamos o seguinte na liña de comandos:

```
java Produto2Interactivo < datos.txt
```

O resultado da execución debe ser a mensaxe: O dobre de 80 e: 160.

Instalación de Apache NetBeans en Linux

Imos instalar Apache NetBeans en Ubuntu.

Para instalar Apache NetBeans en Linux é necesario instalar previamente o **Java JDK**.

Podemos ver de instalar a versión de NetBeans usando un **paquete Snap**. Para facelo:

```
sudo snap install netbeans --classic
```

A finais de outubro de 2021 instala o Apache NetBeans 12.5.

Ubuntu Snap é un novo concepto que apareceu en Ubuntu 16.04. Trátase dunha nova forma de instalar aplicacións en Ubuntu cun paquete snap que contén a aplicación que se quere instalar xunto coas súas dependencias. O problema

evidente que presentan estes snaps é o tamaño que teñen, xa que, ao conter as dependencian ocupan máis espazo en disco.

Se queremos instalar un Netbeans distinto do anterior, podemos seguir os seguintes pasos.

Imos instalar a versión 12.0 xa que se trata da última versión LTS.

Para **instalar Apache NetBeans 12.0**, primeiro descargamos o arquivo binario:

```
wget https://downloads.apache.org/netbeans/netbeans/12.0/netbeans-12.0-bin.zip
```

Unha vez descargado, descomprimimos o arquivo descargado:

```
unzip netbeans-12.0-bin.zip
```

Tras descomprimir movemos o cartafol de netbeans ao cartafol /opt que é o cartafol que se usa en Linux para instalar as aplicación de terceiros:

```
sudo mv netbeans/ /opt/
```

O executable de Netbeans está en /opt/netbeans/bin/netbeans. É necesario engadir o seu directorio pai ao **$PATH** para que podamos lanzar o programa sen especificar a ruta absoluta ao executable.

Cando tentamos executar un comando, o sistema busca o ficheiro executable nunha lista de directorios que está almacenada na variable de contorno chamada $PATH.

Para actualizar a variable $PATH do usuario de linux o que facemos é modificar o arquivo **.bashrc** deste usuario. O arquivo .bashrc é un script que se executa cada vez que un usuario inicia sesión nun terminal de Linux.

Abrimos o arquivo ~/.bashrc:

```
nano ~/.bashrc
```

Engadimos a seguinte liña ao final do ficheiro:

```
export PATH="$PATH:/opt/netbeans/bin/"
```

Para poder lanzar NetBeans sen reiniciar a shell usamos o comando:

```
source ~/.bashrc
```

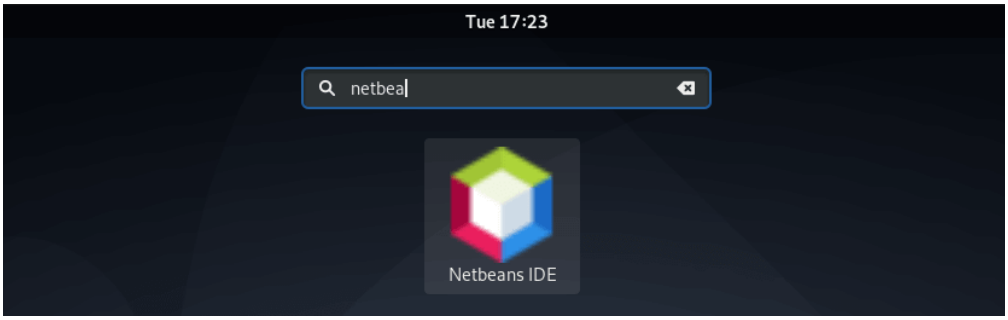
Para crear un arquivo .desktop que permita que a aplicación NetBeans apareza no menú de aplicacións creamos o ficheiro **netbeans.desktop** en /usr/share/applications/:

```
sudo nano /usr/share/applications/netbeans.desktop
```

No ficheiro escribimos os seguintes datos:

```
[Desktop Entry]
Name=Netbeans IDE
Comment=Netbeans IDE
Type=Application
Encoding=UTF-8
Exec=/opt/netbeans/bin/netbeans
Icon=/opt/netbeans/nb/netbeans.png
Categories=GNOME;Application;Development;
Terminal=false
StartupNotify=true
```

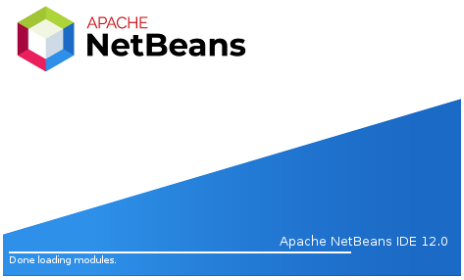
Tras isto, xa temos instalado o IDE NetBeans e podemos lanzalo desde o menú de aplicación.



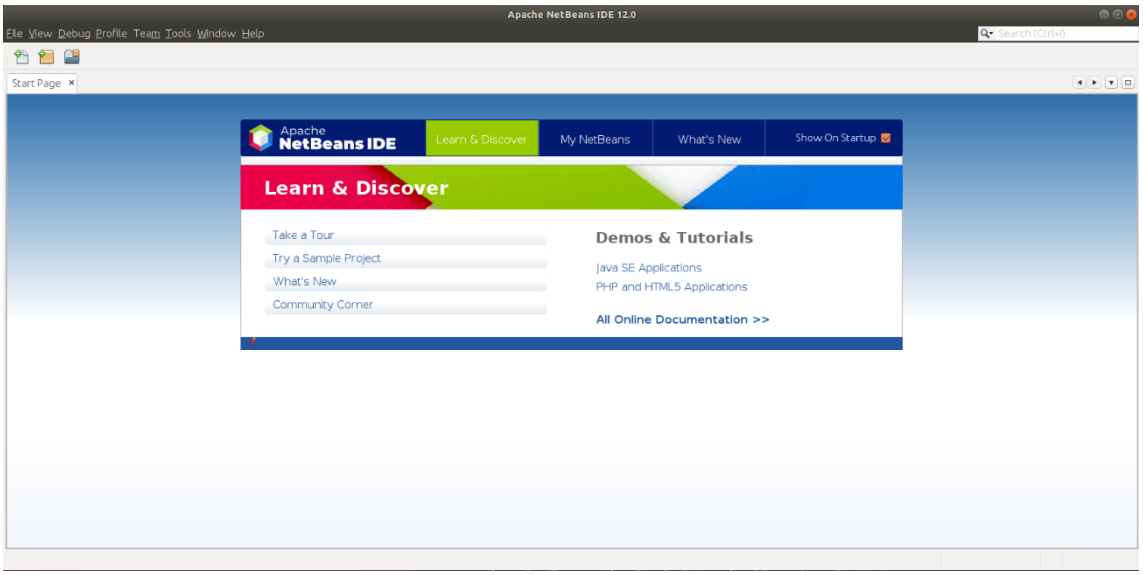
Tamén podemos lanzalo desde o terminal usando o comando netbeans:

```
netbeans
```

Debe mostrarse unha pantalla como a seguinte:



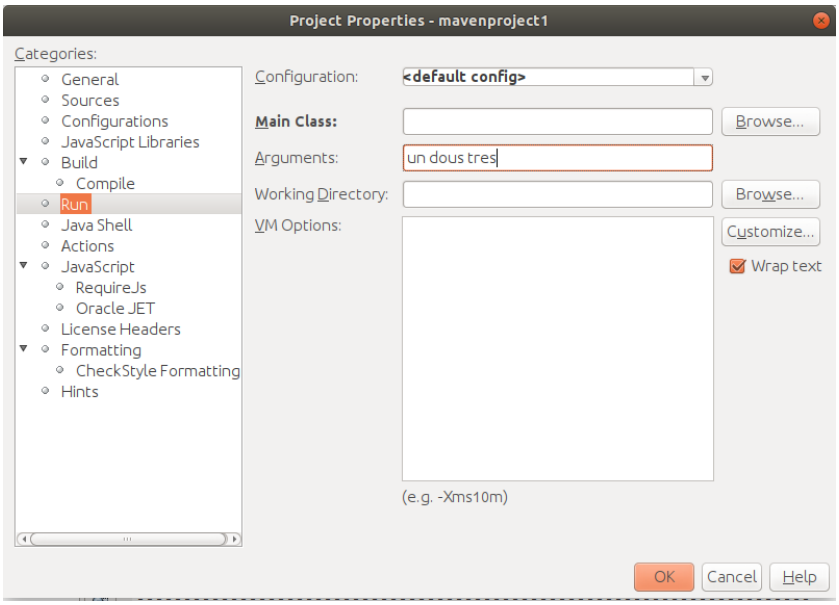
Cando a aplicación se teña iniciado, debería verse a páxina de benvida por defecto de NetBeans.



A partir deste momento, xa podemos crear os nosos **proxectos Java** con NetBeans.

Temos que ter en conta que, os IDE, non traballan con arquivos e cartafois, usan o concepto de **proxecto** en vez do de cartafol. Un proxecto loxicamente é un cartafol no disco duro, pero nun IDE créanse arquivos adicionais ao código para optimizar a experiencia dos usuarios. Nestes arquivos pode estar configuración de execución, despregamento, tipo de proxecto, etc.

Tras crear un proxecto, podemos querer utilizar **argumentos** da liña de comandos. Para facelo imos a **File** eliximos **Project Properties** e en **categories** eliximos **run**. Aparece unha caixa de texto para indicar os argumentos.



Instalación de Apache NetBeans en Windows

Na URL https://netbeans.apache.org/download/index.html pode descargarse Apache Netbeans.

Eliximos a última versión LTS (Apache Netbeans 12.0).

Descargamos o instalador para Windows e instalamos.

Instalación de Visual Studio Code en Linux

Imos facer a instalación en Ubuntu.

Instalamos Visual Studio Code desde o repositorio oficial **VS Code** que nos ofrece **Microsoft**.

Comenzamos actualizando o índice de apt:

```
sudo apt update
```

Continuamos instalando paquetes necesarios:

```
sudo apt install software-properties-common apt-transport-https wget
```

Descargamos e instalamos a clave GPG de Microsoft:

```
wget -q https://packages.microsoft.com/keys/microsoft.asc -O- | sudo apt-key add -
```

Agregamos o novo repositorio:

```
sudo add-apt-repository "deb [arch=amd64] https://packages.microsoft.com/repos/vscode stable main"
```

A instalación é simple:

```
sudo apt update
sudo apt install -y code
```

Tras isto, xa temos Visual Studio Code instalado. Podemos inicialo desde o menú clásico de Linux, ou directamente desde a consola (máis rápido).

Para iniciar desde a consola:

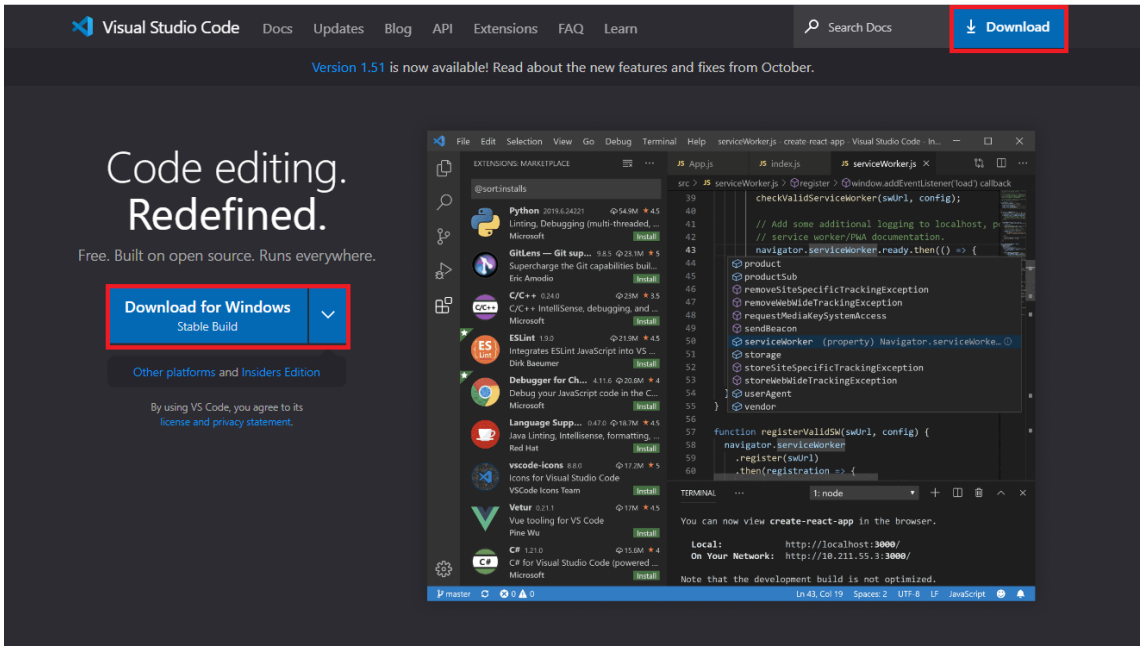
```
code
```

Como norma xeral o editor actualízase cada mes polo que non debemos esquecer executar de vez en cando:

```
sudo apt update
sudo apt upgrade code
```

Instalación de Visual Studio Code en Windows

Desde a páxina oficial de Visual Studio Code (https://code.visualstudio.com/) pode descargarse a última versión estable (mediante o cadro azul grande situado á esquerda) ou acceder á páxina de descargas (mediante o cadro azul máis pequeno situado arriba á dereita).



Desde a páxina de descargas pode descargarse versións para diferentes sistemas operativos (32/64 bits, Windows/Linux/Mac). En Windows hai ademais dispoñibles versións System Installer que se instalar no cartafol de Archivos de programa e versións User Installer que se instalan no cartafol de usuario. Desde o verán de 2018, Microsoft recomenda a versión User Installer.

Unha vez descargado o instalador iniciamos o asistente de instalación facendo dobre clic sobre o mesmo.

A continuación esixe aceptar a licenza de Visual Studio Code para poder continuar coa instalación.

A segunda pantalla permite elixir o directorio de instalación.

A terceira pantalla permite elixir o nome do cartafol do menú de inicio.

A cuarta pantalla permite seleccionar algunhas tarefas adicionais tras a instalación.

Finalmente amósanse as opcións elixidas nas pantallas anteriores. Para iniciar a instalación, facemos clic en instalar. A continuación, instalarase Visual Studio Code. Visual Studio Code actualízase automaticamente cada vez que se publica unha nova versión.

Instalación de plugins en NetBeans

Unha vez baixada e instalada a versión básica do IDE, podemos baixar outros paquetes ou plugins conforme os vaiamos necesitando.

Os plugins para NetBeans poden instalarse desde dúas fontes:

1. O catálogo de plugins de NetBeans que podemos consultar desde o IDE (instalación online).
2. Arquivos .nbm (NetBeans Module) que podemos atopar nalgunhas páxinas de internet, principalmente desde o portal de plugins de NetBeans (https://plugins.netbeans.apache.org - instalación offline).

Instalación desde o catálogo de plugins

Esta forma de instalación é a máis simple e directa. Permítenos desde a mesma interface de NetBeans, revisar un catálogo de plugins e descargalos. Tamén é a máis segura xa que no caso de que o plugin que estamos tratando de instalar necesite outros plugins, estes plugins tamén son descargados e instalados. Ademais, podemos estar seguros de teñen pasado por un proceso de verificación e non debe haber ningún problema ao instalalos.

Para acceder a este catálogo, dirixímonos ao menú **Tools** -> **Plugins**. Con isto, aparecerá a ventá de **Plugins**. Esta ventá ten 5 pestanas:

- As actualizacións atopadas dos plugins que xa temos instalados.
- Os plugins dispoñibles e que non temos instalado.
- Os plugins que descargamos (arquivos .nbm) que queremos instalar.
- Os plugins que xa temos instalado.

Podemos ver de instalar desta forma o plugin **Rainbow Braces** que pon cores diferentes a cada par de corchetes do código.

Para descargar e instalar os plugins seleccionados, facemos click no botón **Install** co que se nos amosará a lista de plugins que serán instalados. Facemos click en **Next** e mostrará un acordo de licenza. Se aceptamos o acordo, comezará a descarga e, posteriormente, realizarase a instalación.

Posteriormente, comprobaremos que xa podemos usar o novo plugin.

Instalación desde arquivos .nbm

Esta forma de instalación ten a vantaxe de que non é necesario que esteamos conectados a internet para realizala xa que toda a información do plugin está no arquivo .nbm.

Non obstante, se un plugin necesita doutros para funcionar, será necesario que nos conectemos a internet, ou ben, que contemos co arquivo .nbm dese outro plugin.

Estes arquivos poden descargarse desde diversos sitios de internet. Normalmente farémolo desde o portal de plugins de NetBeans. Neste portal podemos ver os últimos plugins subidos, os plugins máis populares, os mellor clasificados, por categorías ou podemos buscalos por algún criterio.

Para facer probas, podemos instalar o **Netbeans Case Converter**. Para isto, imos á páxina de plugins de NetBeans e descargamos este plugins. Podemos descargar o arquivo en https://plugins.netbeans.apache.org/catalogue/?id=50.

Tras descargar fr-evidev-netbeans-caseconverter.nbm, no NetBeans, accedemos ao cartafol de descargas e seleccionamos o arquivo .nbm. Unha vez seleccionado o arquivo xa podemos realizar a instalación.

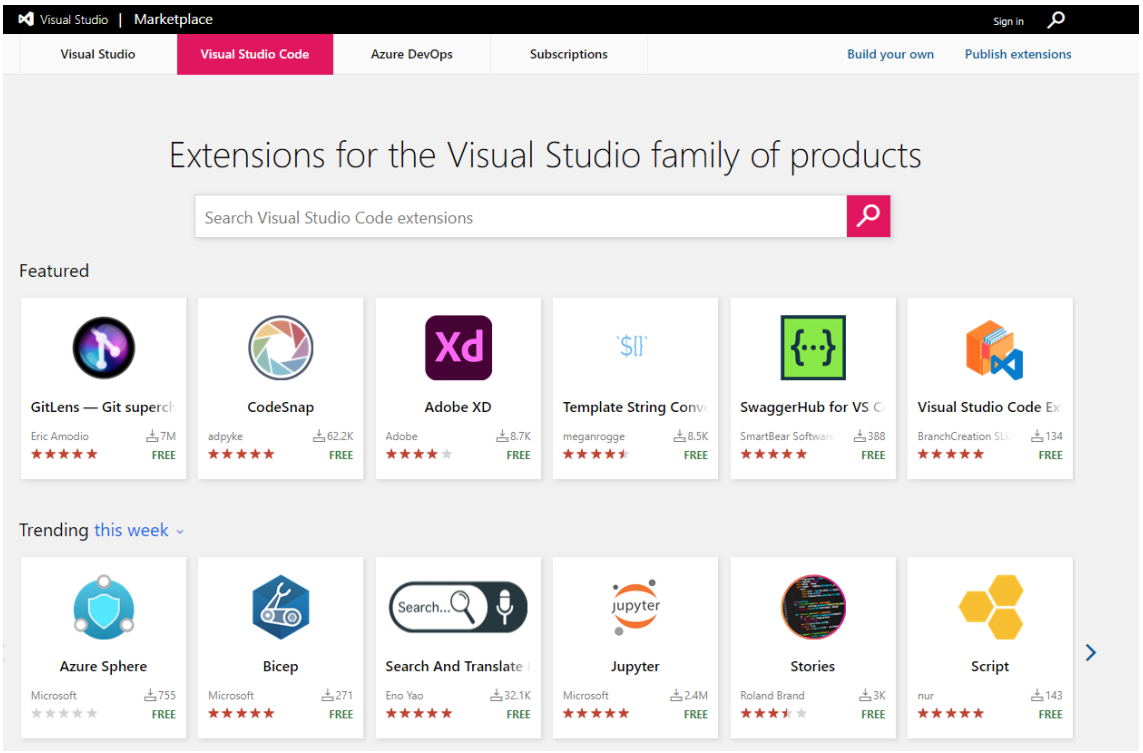
Tras reiniciar o IDE, xa podemos probar o plugin.

Instalación de extensións en Visual Studio Code

As extensións son paquetes de código que se executan dentro de Visual Studio e que ofrecen características novas ou melloradas.

Á hora de instalar extensións nesta ferramenta podemos facelo de dúas maneiras diferentes. A primeira delas é desde a tenda de extensións de Visual Studio Code: https://marketplace.visualstudio.com/vscode.

Unha vez dentro da tenda, simplemente debemos buscar a extensión que vaiamos utilizar para, posteriormente pulsar sobre o botón Install. En Windows a web tentará abrir o editor de texto para proceder a instalar automaticamente a extensión nel polo que, se non temos instalado VSC aínda, teremos que instalalo antes de instalar extensións. En Linux, a web dinos de lanzar VS Code Quick Open (Ctrl+P), pegar o comando e pulsar enter.



Microsoft Visual Studio Code tamén ten integrado un buscador vinculado á tenda desde o que podemos buscar directamente a extensión que necesitemos. Para facelo, pulsaremos o botón Extensions (ou o atallo de teclado control+Shift+X) e desde o cadro que nos aparece podemos buscar a extensión que queiramos.

En caso de que unha extensión xa non nos guste ou non necesitemos usala máis ou non queiramos tela instalada por calquera motivo, podemos **deshabilitala** ou incluso **desinstalala** sen problemas e de forma sinxela.

Aínda que non é obrigatorio, unha vez desinstalada ou deshabilitada unha extensión, o mesmo que tras instalar unha, o mellor é reiniciar o IDE para que todo volva cargarse con normalidade.

Javadoc

Javadoc é unha ferramenta do SDK que permite documentar, dunha maneira rápida e sinxela, as clases e métodos que se crean, sendo de grande utilidade para a comprensión do desenvolvemento.

A linguaxe Java soporta tres tipos de comentarios:

/* texto */. O compilador ignora todo o que estea comprendido entre /* e */.

// texto. O compilador ignora todo o que estea a partir de // ata a fin da liña.

/** documentacion */. Este é un comentario de documentación. A ferramenta javadoc do JDK usa os comentarios de documentación para preparar a documentación que se xera automaticamente.

Para usar en liña de comandos javadoc hai moitos parámetros posibles. Se queremos crear un cartafol de nome probaDoc e almacenar nel a documentación sobre o arquivo de código proba.java podemos facer:

```
javadoc -d ./probaDoc/ Proba.java
```

Para visualizar a documentación xerada podemos facer:

```
firefox ./probaDoc/index.html
```

A documentación que se xera ten o seguinte aspecto: