El lenguaje Ruby

En este primer capítulo nos introduciremos en el lenguaje, conoceremos sus cualidades y veremos por qué Ruby es una alternativa cada vez más interesante, tanto como lenguaje orientado a proyectos web como para realizar desarrollos de aplicaciones de escritorio.

BREVE INTRODUCCIÓN A LA PROGRAMACIÓN

Antes de comenzar con un lenguaje específico, siempre es conveniente familiarizarnos con los conceptos básicos. La razón principal para aprender un lenguaje y programar es utilizar la computadora como una herramienta para resolver problemas. Toda resolución de un problema supone una serie de fases o pasos, entre los cuales podemos encontrar los siguientes:

- Análisis del problema
- Diseño del algoritmo
- Conversión del algoritmo a un programa
- Ejecución del programa

¿Qué es un algoritmo?

La palabra **algoritmo** proviene de Mohammed al-Khowârizmî, matemático persa que vivió durante el siglo IX. Este matemático enunció un conjunto de reglas determinadas con el fin de realizar las operaciones básicas de suma, resta, multiplicación y división. Más tarde, el apellido fue traducido al latín, y la palabra algoritmo proviene de allí.
Básicamente, un algoritmo es un método para resolver un problema.
Decimos que es un conjunto finito de operaciones bien definidas y ordenadas que permiten hallar la solución a un problema.
Esta lista de pasos para la resolución es luego transferida, en nuestro caso, a un conjunto de instrucciones capaces de ser analizadas y ejecutadas por un procesador o automatizadas de alguna forma dada. La traducción a instrucciones será hecha en algún lenguaje particular de programación.

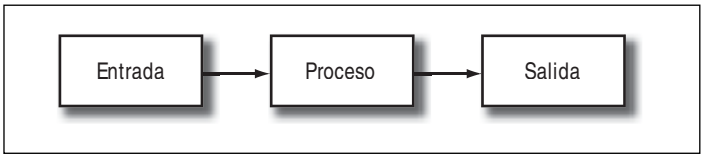

Figura 1. *Podemos observar cómo se representa un algoritmo en un alto nivel de abstracción.*

Cuando trabajamos con algoritmos, debemos ser conscientes del nivel de complejidad con el que operaremos. Dado el mismo problema, es posible encontrar soluciones de distinto nivel de complejidad; por lo tanto, se recomienda siempre mantener el foco en la solución real.

Características de los algoritmos

Todo algoritmo debe cumplir con las siguientes características:

• debe ser preciso y ordenado en cuanto a los pasos por ejecutar;
• debe estar definido para obtener el mismo resultado siguiendo el mismo camino;
• debe ser finito.

Como en otras disciplinas, se dice que un algoritmo es eficiente cuando logra llegar a sus objetivos planteados utilizando la menor cantidad de recursos posibles, mientras que un algoritmo es eficaz cuando alcanza el objetivo primordial.
Un algoritmo es descrito mediante tres partes esenciales: **entrada**, **proceso** y **salida.**

¿Cómo se representan los algoritmos?

Para escapar de la ambigüedad presente en el idioma natural, en informática utilizamos gran cantidad de herramientas para modelar o representar los algoritmos que diseñamos para la resolución de problemas. Para presentarlo de una manera simple, decimos que existen dos técnicas: modelos gráficos y modelos textuales.

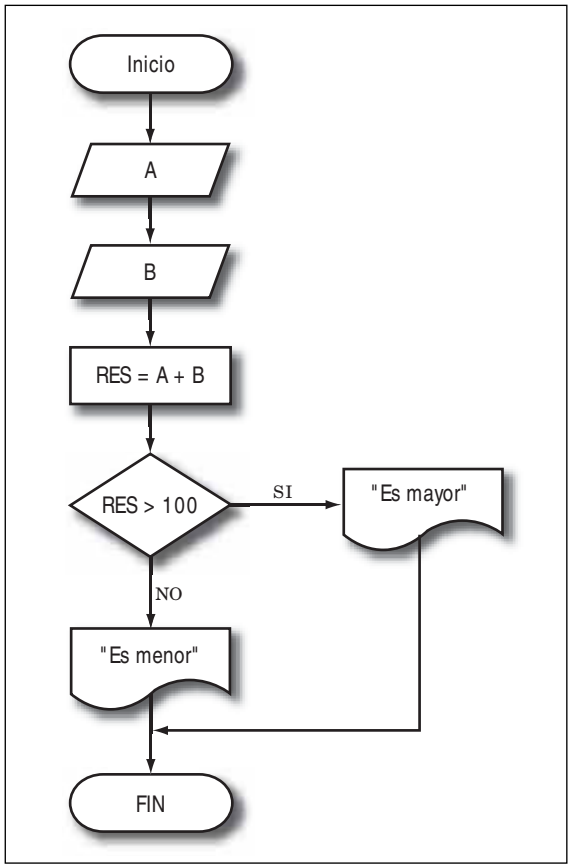

Figura 2. *Una forma de representar una solución es utilizar diagramas de flujo.*

Los modelos textuales, aunque son frecuentemente usados, suelen presentar ciertas deficiencias, que debemos saber:

- lenguaje no específico del problema;
- ambigüedad;
- dificultad para representar conceptos de forma clara y simple.

Debido a algunos de estos inconvenientes, también se utilizan (y son recomendables) las técnicas gráficas. Los gráficos permiten que el equipo de desarrollo pueda tener un lenguaje común y que entiendan el problema de un solo pantallazo. Existen muchos modelos, y una parte esencial de todo proyecto consiste en decidir qué herramientas de modelado se utilizarán.

¿Qué son los lenguajes de programación?

Básicamente, son lenguajes que permiten controlar el comportamiento de una máquina. Como todo lenguaje, están integrados por un **conjunto de reglas semánticas y sintácticas**. Poseen una estructura determinada, elementos y expresiones. Todo lenguaje de programación tiene instrucciones que pueden ser divididas en cuatro grupos:

- instrucciones de entrada y salida;
- instrucciones aritméticas y lógicas;
- instrucciones selectivas;
- instrucciones repetitivas.

Existen distintos tipos de lenguajes, pero la división básica se hace de acuerdo con el nivel de abstracción en el que se encuentran las sentencias. Es así como tenemos:

- **Lenguajes máquina**: aquellos directamente inteligibles por la máquina. Sus instrucciones son cadenas binarias, que especifican una operación y las direcciones de memoria necesarias. Un hardware posee sus propias instrucciones de lenguaje máquina.

REPRESENTAR ALGORITMOS

Muchas técnicas y lenguajes nos permiten representar de manera fiel y agradable los algoritmos, sean éstos sencillos o complejos. Entre los diagramas básicos que no podemos desconocer, al menos por su valor histórico, están los **diagramas de flujo** y la **diagramación Jackson**. Si queremos representar un algoritmo en un nivel de abstracción menor, podemos utilizar pseudocódigo.fdgbdfghfg

- **Lenguajes de bajo nivel**: poseen instrucciones mnemotécnicas, por ejemplo: ADD, SUB, DIV. El lenguaje de bajo nivel clásico es el **ensamblador**. Estos lenguajes también dependen del hardware.
- **Lenguajes de alto nivel**: son diseñados de forma tal que los programadores sean quienes puedan interpretar y leer el lenguaje de forma natural. Además, no dependen de hardware particular.

Podemos suponer que cada uno de estos tipos de lenguaje tiene inconvenientes y ventajas. En general, las ventajas de los lenguajes de alto nivel son:

- curva de aprendizaje menor;
- sintaxis y semántica similar a los lenguajes humanos;
- reducción de tiempos de desarrollo;
- reducción de costos;
- transportabilidad.

Tenemos como contrapartida, que se necesita más capacidad de memoria, y el tiempo de ejecución es mayor.

Traductores de lenguajes

Vimos que, con los lenguajes máquina, hablamos el idioma de las computadoras mientras que, al utilizar otro tipo de lenguaje que nos facilita la tarea, dejamos de entendernos con ella. Para que la comunicación pueda ser posible, necesitamos de uno o más traductores. Estos aplicativos traducen los programas escritos en lenguajes de medio y alto nivel, a lenguaje máquina. Existen básicamente dos tipos de traductores, que se separan en:

Compiladores: son aplicaciones que traducen el código fuente de un programa a otro lenguaje de nivel inferior.
Intérpretes: los intérpretes también hacen una traducción a código máquina con la diferencia que es realizada línea a línea.

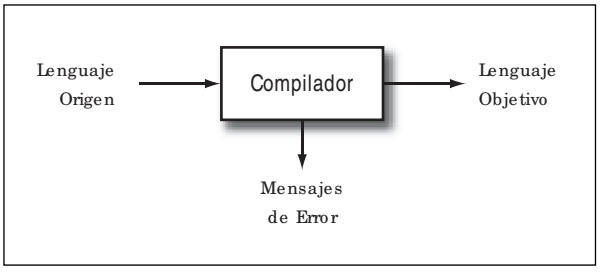

Figura 3. *Representación gráfica de un compilador, donde se omiten las etapas intermedias.*

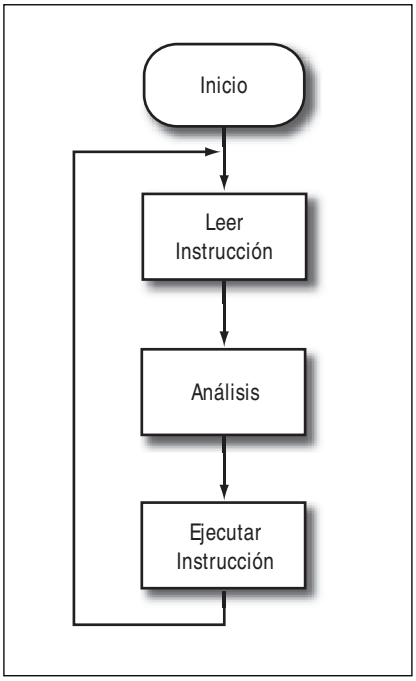

Figura 4. *Representación gráfica de un intérprete genérico, similar al utilizado por Ruby.*

Diferencias entre compiladores e intérpretes

Una vez que un programa ha sido compilado, se transforma en otro lenguaje, a diferencia del intérprete, que traduce el programa cada vez que se lo ejecuta. La principal ventaja en la que está basada el lenguaje intérprete, reside en la portabilidad, ya que permite que el mismo programa pueda ser trasladado a otras plataformas, mientras que el archivo generado por el compilador sólo es funcional en la plataforma de origen. La desventaja del intérprete es, por lo general, su velocidad bastante menor que la de los aplicativos compilados, ya que debe pasar por varias etapas o capas, para que se comprendan todas sus instrucciones. Actualmente, existen algunos esquemas mixtos que comparten características de los dos "mundos". En la **Figura 5** observamos las etapas de la compilación.

III LENGUAJES DE PROGRAMACIÓN

Existe una rama de la informática denominada Teoría de los lenguajes de programación, más reconocida por sus siglas **PLT**. Además de estudiar la historia y la evolución de los lenguajes, se interesa por áreas de diseño y desarrollo, principalmente orientadas a conocer las capacidades y características de todos los lenguajes disponibles, que actualmente son más de 500.

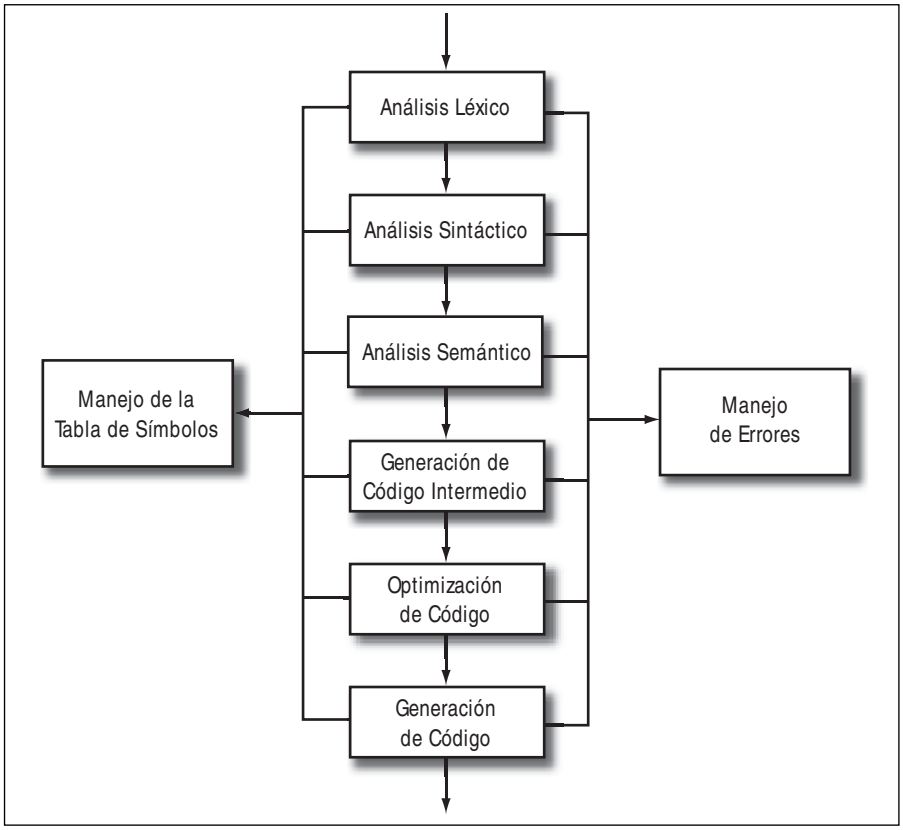

Figura 5. *Podemos observar la cantidad de operaciones que se realizan en el proceso llamado compilación.*

PROGRAMAR EN RUBY

Para comenzar a programar, sólo nos hace falta una computadora, algunos conceptos básicos de software y hardware, y muchas ganas. La elección de un lenguaje puede ser una dificultad, pero en este caso hemos seleccionado **Ruby** y trabajaremos con él. No debemos dejar nunca de lado los conceptos teóricos que aprenderemos, puesto que nos servirán no sólo para un lenguaje en particular, sino para poder pasar de un lenguaje a otro teniendo sólo los problemas de sintaxis específicos y no los baches de nivel lógico. Ya estamos listos, comencemos a conocer Ruby.

Introducción a Ruby

Actualmente, existe una gran cantidad de lenguajes de programación que son utilizados en diversas áreas, desde el desarrollo de aplicaciones administrativas hasta el

campo de la inteligencia artificial. Es muy complicado conocer a fondo las posibilidades que nos presenta cada uno de estos lenguajes y, por lo tanto, a la hora de seleccionar, lo hacemos sobre la base de nuestros gustos o inquietudes. Ruby se presenta como un lenguaje **sencillo** y **flexible** que atrae a programadores de todos los sectores y que promete una grata experiencia en el trabajo habitual.
A pesar de tener muchos años en el mercado, el auge del lenguaje llegó de la mano de un framework para aplicaciones web denominado **Rails**. Esto hizo que muchos desarrolladores web migraran desde sus lenguajes más tradicionales, como **PHP** o **ASP**, a la nueva y fascinante opción. Sin embargo, Ruby es un lenguaje multipropósito que permite desarrollos en las siguientes áreas:

- aplicaciones comerciales;
- acceso a base de datos;
- proceso y transformación de XML;
- aplicaciones distribuidas;
- aplicaciones web.

Historia del lenguaje

Ruby fue creado en el Japón por **Yukihiro Matsumoto** mientras trabajaba como programador con lenguajes como Perl y PHP. En principio, su intención fue la de crear un Perl avanzado debido a que deseaba mejorar algunas de las preciadas particularidades de este conocido lenguaje. Pero en lugar de mejorarlo, se vio tentado a desarrollar uno propio a partir de sus lenguajes preferidos: **Perl**, **Smalltalk**, **Eiffel** y **Lisp**. De esta forma surge el lenguaje Ruby, aunque en ese momento aún no contaba con ninguna línea de código. Luego de más de dos años de trabajo, Ruby se presenta al público en su versión 0.95. En esta etapa, todo lo relacionado con el lenguaje era precario y todavía no contaba con gran empuje; tanto es así que se anuncia que el CVS sería lanzado semanas después. Finalmente, en 1996, Ruby 1.0 es ofrecido al público. A partir de 1997, varias empresas se interesan en Ruby como un campo para explorar, y ese mismo año se escribe el primer artículo técnico. Un año después, aparece la página oficial en idioma inglés;

{} EL NOMBRE RUBY

Según el creador de Ruby, Yukihiro Matsumoto, el nombre del lenguaje lo decidió en honor a un colega suyo a partir de la piedra correspondiente a su mes de nacimiento. También existe un juego de palabras relacionado con el lenguaje Perl (preferido de Matsumoto), ya que en un principio su idea fue la de crear una versión de este lenguaje mejorado.

empiezan a hacerse charlas y conferencias sobre el lenguaje, con gran aceptación en los ambientes académicos. En el año 2000, IBM se interesa en el lenguaje y publica un artículo acerca de la denominada *Latest open source gem from Japan* (La última gema del open source del Japón). El lenguaje creció de forma lenta, pero sostenida, hasta el 2004, cuando Rails fue liberado. David Heinemeier Hansson crea este framework cuya primera versión (1.0) salió definitivamente un año después. A partir de la aparición de Rails, el crecimiento de Ruby ha sido extraordinario: se lo ha seleccionado como **el lenguaje de programación del 2006** y se encuentra entre los 10 más populares de la actualidad según el ranking **TIOBE**.

Figura 6. *En la página oficial de Ruby, encontramos una sección dedicada a la historia y a su creador, desde la cual también podremos descargar la versión actual del lenguaje.*

¿Por qué usar Ruby?

A continuación, describimos algunas características particulares de este lenguaje, que indican sus ventajas y beneficios para su implementación:

- Ruby es un lenguaje de **scripts**, **moderno** y **orientado a objetos**, que combina una importante flexibilidad con alta productividad.
- Incorpora algunas de las mejores características de otros lenguajes como Smalltalk, Java y Perl.
- Su alcance parece ilimitado y hoy se encuentra presente en aplicaciones que van desde el desarrollo web hasta la simulación de ambientes complejos.

- Es un lenguaje **multiplataforma** que se integra perfectamente en gran cantidad de arquitecturas; puede correr, incluso, en dispositivos móviles.
- Promueve las mejores prácticas de programación sin perder usabilidad.
- Mediante su uso se pueden complementar las características de la lógica imperativa con la lógica funcional.
- Es altamente extensible no sólo mediante librerías escritas en Ruby, sino que podemos ampliarlo utilizando el lenguaje C y, actualmente, de forma experimental otros lenguajes.
- Posee una filosofía real de trabajo, que propone algunas prácticas particulares como **DRY** (*Don´t repeat yourself*; en español: No te repitas) entre otras.
- Simplifica declaraciones, estructuras y modelos sin perder potencia y permite que el programador, se desarrolle de forma adecuada.
- Es un lenguaje **dinámico** e **interpretado**, con las características de éstos.
- Permite utilizar la más **simple expresión** para un programa o algoritmo; esto sumado a las actuales prácticas ágiles permite desarrollar en forma amigable.

Si aún, a pesar de lo mencionado, existen dudas para elegir a Ruby sirve aclarar que grandes empresas y usuarios empezaron a desarrollar **proyectos** para utilizar a Ruby en sus arquitecturas o entornos. Actualmente, se destacan dos proyectos **Ruby.Net** y **JRuby.** Cada uno busca interoperatividad total entre plataformas.

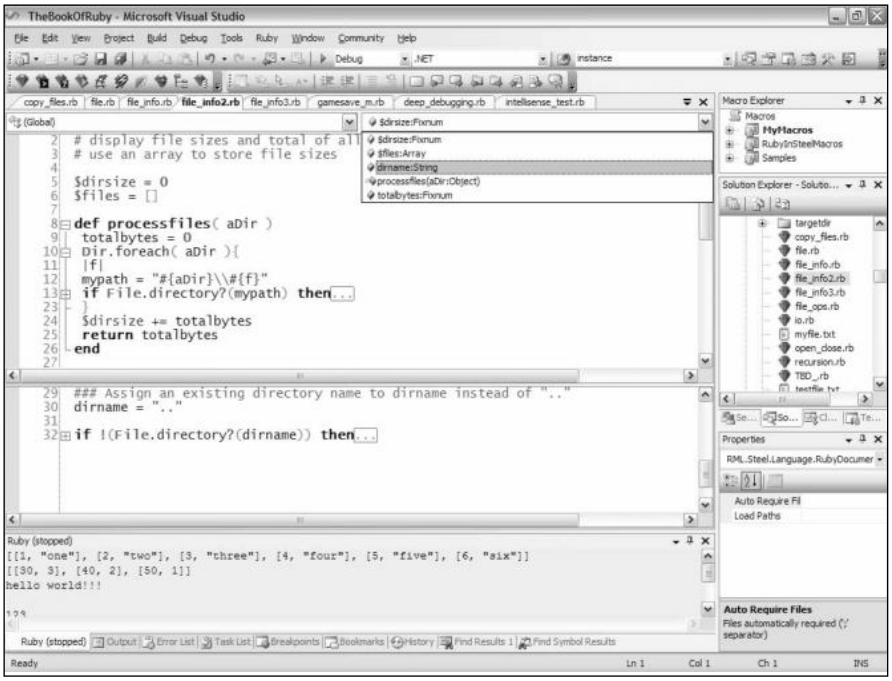

Figura 7. *Apreciamos cómo puede programarse en Ruby en distintas plataformas y ambientes de desarrollo. En este caso,* ***Visual Studio****.*

Por último, es importante destacar que el desarrollo con Ruby resulta entretenido, sencillo y simple. Experimentar con Ruby permite que nuestro trabajo sea más agradable y, por lo tanto, más satisfactorio.

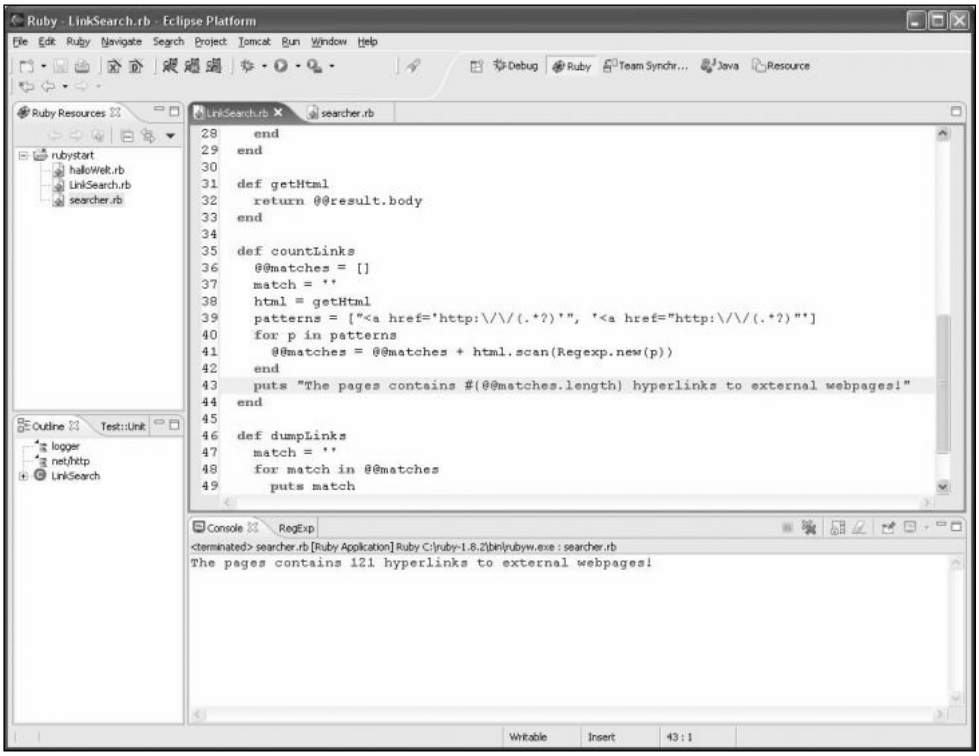

Figura 8. *Podemos desarrollar cómodamente en el entorno* ***Eclipse****.*

CARACTERÍSTICAS DEL LENGUAJE

Ya hemos presentado el lenguaje y su historia; es tiempo de que veamos sus características y de ejemplificar algunas de ellas.

HERENCIA MÚLTIPLE

Es una realidad que en muchos lenguajes modernos no existe la herencia múltiple; esto suele ser debido a que, generalmente, se presentan problemas de jerarquía. Encontramos muchos lenguajes que posibilitan el uso de herencia múltiple, entre ellos, el más conocido es sin lugar a dudas **C++.**

- **En Ruby todo es un objeto**: esto básicamente quiere decir que desde el más simple carácter hasta un conjunto de instrucciones, son **instancias de clases** y serán manipuladas como tales. Este concepto anula lo que normalmente denominamos tipos primitivos, ya que hasta el más trivial de los datos es un objeto.
- La **gran flexibilidad** de Ruby permite que se pueda incorporar funcionalidad en sus clases base y en sus métodos. Es decir, podemos **modificar absolutamente todo** dentro del ambiente.
- En el lenguaje, todo tiene un valor, aunque sea **nil.**
- Debemos saber que, en principio, **no existen diferencias entre comandos y expresiones** dentro del entorno de programación.
- Ruby utiliza sólo herencia simple. Esta característica habitual en muchos lenguajes facilita el trabajo con estructuras jerárquicas. Sin embargo, incorpora técnicas para poder imitar el comportamiento de la herencia múltiple de manera más sencilla. Éstas las veremos más adelante, pero comprenden el uso de módulos y **mixin.**
- Ruby utiliza un recolector de basura de alto nivel. Por lo tanto, libera al desarrollador de estas tareas, en algunos casos triviales.
- **No es de tipo estricto** y no requiere declaración de variables.
- Ruby permite la programación con **múltiples hilos** de forma independiente al sistema operativo.

RUBY DESDE OTROS LENGUAJES

Seguramente, cuando elegimos un nuevo lenguaje, nos interesa saber qué características en particular lo asemejan o lo diferencian del lenguaje en el cual estamos desarrollando actualmente, y con el que ya estamos familiarizados. Para esto, proponemos las comparaciones que se encuentran a continuación.

Hacia Ruby desde C

Comencemos diciendo que Ruby está enteramente desarrollado en C. Las similitudes que posee Ruby con C no son demasiadas en la flexibilidad de trabajo que obtenemos con él, sin embargo, podemos enumerar algunas: la mayoría de los operadores, el tratamiento de algunas cadenas y la sensación de estar a cargo de lo que se desarrolla.

Entre otras cosas, como Ruby es interpretado, debemos esperar tiempos de respuesta bastante mayores que los que obtenemos con C. Sin embargo, gozamos de algunas mejoras como el excelente recolector de basura y la posibilidad de seguir operando de forma procedimental. Esta opción no evita que de fondo estemos

trabajando con objetos. Otra diferencia con respecto a C son las posibilidades que se nos brindan para el desarrollo web.
Una característica deseada para los desarrolladores de C es que Ruby resulta fácilmente **extensible** a partir de **módulos** que pueden estar enteramente desarrolladas en C a pesar de actuar como si lo estuvieran en Ruby. Esta particularidad, sumada a la facilidad con que podemos solucionar problemas de software, hace de Ruby una excelente elección.

Hacia Ruby desde Java

Dado que Ruby actualmente cuenta con una creciente popularidad, es común que se discutan las ventajas y desventajas frente a otros lenguajes mejor posicionados como **Java** o **C#**. Los seguidores de éstos marcarán como primera falencia de Ruby su velocidad; y es cierto que en este aspecto todavía queda mucho campo por recorrer. Pero debemos destacar también que es el mismo motivo que utilizaría un desarrollador de C++ para desacreditar a esos lenguajes.
Una vez mencionado este aspecto; podemos decir que todo desarrollador de Java se encontrará con muchas similitudes, como la utilización de objetos de tipado estricto; la existencia de métodos públicos, privados y protegidos; utilización de **Rdoc** para la documentación (similar a javaDoc).

Entre las diferencias podemos marcar: la necesidad de métodos para acceder a todo (las variables de instancia son privadas); todo es un objeto, y no existe la declaración de tipo de datos.

Hacia Ruby desde Visual Basic

Esta comparación entre lenguajes diametralmente opuestos es útil para conocer que Ruby puede ser, en algunos casos, tan productivo o más que VB, que con sus accesibles herramientas y su entorno es un completo RAD. Las prácticas de Ruby y sus concesiones le serán familiares a los desarrolladores de VB, que obtendrán una curva de aprendizaje similar.

DESARROLLADORES DE JAVA Y RUBY

Gran cantidad de desarrolladores Java se han visto sorprendidos gratamente con las cualidades de Ruby. Ruby les ofrece interesantes características con prácticas similares y más sencillas. Existen varios proyectos para permitir la interacción entre los dos lenguajes. El que se destaca entre ellos es **JRuby**.

¿QUÉ PODEMOS HACER CON RUBY?

Hemos mencionado que a partir de Ruby podemos desarrollar todo tipo de aplicaciones. También es cierto que, si recurrimos a los buscadores más populares, veremos que el impacto real del lenguaje se da en el ámbito web. A continuación, mencionaremos algunos proyectos de distinto tipo para poder observar el alcance del lenguaje, aunque pronto veremos que no se agota aquí.

Proyecto Basecamp

Basecamp es un gestor y organizador para trabajo en equipo. Está catalogado como una de las mejores opciones para la colaboración entre equipos o personas. Resulta un gran proyecto que utiliza Rails como arquitectura y es de los más representativos de la arquitectura Rails.

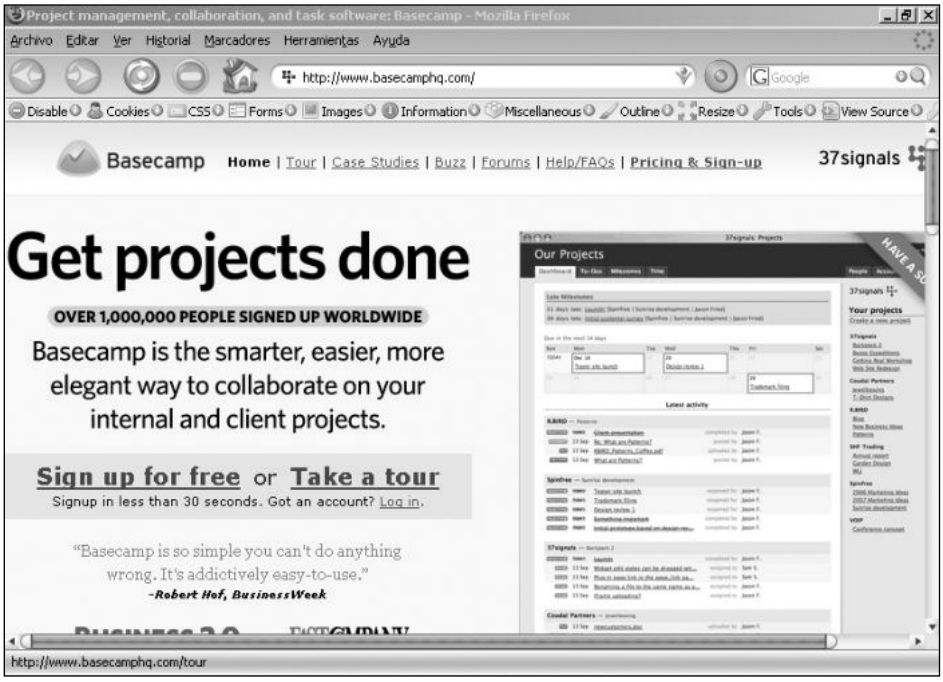

Figura 9. *En la página oficial del proyecto Basecamp, podemos observar la cantidad de proyectos disponibles.*

Proyecto Odeo

Se trata de una red social basada en el intercambio de música. Este gran portal está enteramente desarrollado con Rails y aloja más de 1000 canales de música y más de 1.000.000 de archivos de audio. Sigue en crecimiento.

Figura 10. *En la página principal de Odeo, encontramos toda la información relativa al sitio.*

Proyectos de escritorio

En este caso, veremos algunas imágenes de proyectos de Ruby haciendo uso de librerías para generar GUI, para aplicaciones de escritorio multiplataforma.

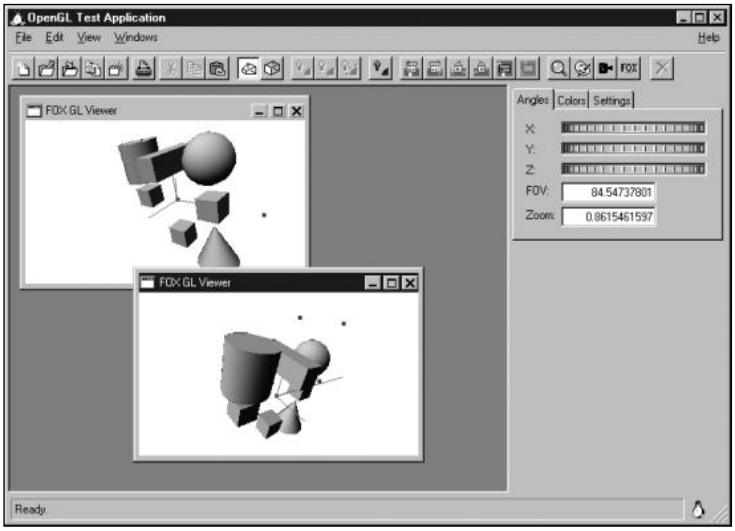

Figura 11. *Aplicación generada con Ruby y un toolkit gráfico desde la línea de comandos sin necesidad de otro aplicativo.*

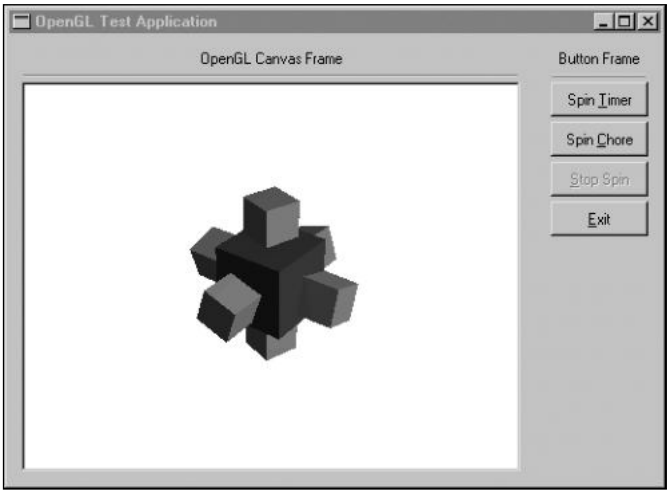

Figura 12. *Interfaz gráfica generada mediante código; hace uso de tecnologías extendidas como OpenGL.*

Existen gran cantidad de comunidades que desarrollan listados de actualización frecuente que nos muestran los distintos proyectos que existen y que utilizan alguna de las tecnologías relacionadas con Ruby.
Una de las más importantes es **RubyForge** (**http://rubyforge.org/**).

Figura 13. *Lista de proyectos open source con Ruby. Actualmente podemos encontrar más de 1000 proyectos activos.*

TECNOLOGÍAS RELACIONADAS

En la actualidad, existe cada vez más la tendencia a integrar herramientas y plataformas. Los lenguajes de programación deben dotarnos de características de avanzada, capaces de facilitar nuestro trabajo. Ruby se integra a la perfección con las últimas tecnologías, como bases de datos, XML, HTML y distribución de paquetes, entre otras. Dos tecnologías o herramientas que están íntimamente ligadas con Ruby son **RubyGems** y **Rails**. Aunque en capítulos posteriores las detallaremos, vale la pena tener un acercamiento a éstas y conocer sus características.

Ruby Gems

Existen sistemas operativos, aplicaciones y herramientas que permiten incorporar o quitar funcionalidad a partir de paquetes que se distribuyen generalmente a través de Internet. Ruby utiliza un gestor de paquetes denominado **RubyGems**. RubyGems proporciona un formato estándar y autocontenido (**gem**) con el objetivo de distribuir programas o librerías en Ruby. Además, tiene herramientas para gestionar la instalación y un servidor para la distribución. Entre sus funciones principales se destacan:

- Instalar los paquetes a distancia.
- Administrar a distancia.
- Administrar dependencias.
- Desinstalar de forma fácil.

Ruby on-rails

Como mencionamos más de una vez, gran parte del éxito de Ruby se debe a Rails. Muchos de ustedes ya estarán preguntándose qué es Rails. Rails o **RoR** (Ruby on Rails) es simplemente un framework en Ruby para aplicaciones web. Rails sigue el paradigma de arquitectura Modelo-Vista-Controlador (**MVC**). Utiliza características avanzadas de Ruby como la **metaprogramación** para facilitar el desarrollo.

METAPROGRAMACIÓN

La metaprogramación consiste en escribir programas que utilizan o manipulan otros programas (o a sí mismos) como datos, o que hacen en tiempo de compilación parte del trabajo que, de otra forma, se haría en tiempo de ejecución. Esto permite al programador realizar más rápido la producción de código. La metaprogramación se encuentra dentro de las técnicas modernas de desarrollo.

Sabemos que en el mercado existen cientos, por no decir miles, de frameworks que facilitan la programación orientada a la red, pero lo que hace a Ruby tan especial es su filosofía. Rails no se diferencia de Ruby en simplicidad y gusto. Rails es distribuido a través de RubyGems, que es el formato oficial del paquete y canal de distribución de librerías y aplicaciones Ruby.

SOFTWARE LIBRE

Dijimos que Ruby es libre, pero es difícil poner en palabras todo lo que significa que un software sea libre y el impacto que produce. Realmente, podríamos obviar esta explicación, pero es importante conocer los principios básicos de este movimiento y cómo nos afectan en nuestras tareas con el lenguaje y nuestras herramientas. Según la definición del proyecto GNU, **www.gnu.org**, el software libre brinda "La libertad a los usuarios de ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software". De modo más preciso, se refiere a cuatro libertades de los usuarios del software:

1. "La libertad de usar el programa, con cualquier propósito (libertad 0)".
2. "La libertad de estudiar cómo funciona el programa, y adaptarlo a tus necesidades (libertad 1). El acceso al código fuente es una condición previa para esto".
3. "La libertad de distribuir copias, con lo que puedes ayudar a tu vecino (libertad 2)".
4. "La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie. (libertad 3). El acceso al código fuente es un requisito previo para esto".

Sin entrar en mayores detalles, debemos aclarar que existen muchas licencias, cada una con sus atributos, que entran dentro de lo que definimos como software libre. En el caso de una herramienta, las ventajas de ser libre son interesantes. Solo imaginemos que podemos tener acceso al código fuente y modificar a nuestro gusto cualquiera de sus partes o que podemos conocer a fondo y desde adentro cómo operan. A su vez, esto hace que ninguna persona pueda privarnos de darle el uso que creamos conveniente.

SOFTWARE LIBRE

A pesar de que habitualmente utilizamos los términos *free software* (software libre) y *open source* (código abierto) para describir lo mismo, cabe aclarar que existen pequeñas diferencias y que se prefiere, en algunos casos, la primera forma por ser más exacta en cuanto a la filosofía real del movimiento.

PROBAR RUBY

Una alternativa excelente a la hora de probar Ruby sin necesidad de instalar nada es a partir de un intérprete interactivo que corre en el navegador web. Esta opción también permite introducir a nuestros colegas en este fantástico mundo.

Figura 14. *En* ***http://tryruby.hobix.com****, encontramos un intérprete interactivo que nos permitirá comenzar a familiarizarnos con Ruby.*

Este intérprete es Ruby 100% y permite que no sólo conozcamos las instrucciones básicas, sino que además podamos seguir varios tutoriales online y, con éstos, aprender los conceptos y la filosofía de Ruby. Pasando algunos minutos con este intérprete, podremos aprender lo básico de manera interactiva.

INSTALAR RUBY

Antes de comenzar a trabajar, debemos instalar Ruby en nuestro entorno. Como sabemos, Ruby es un lenguaje multiplataforma; por lo tanto, puede ser instalado y utilizado en distintos ambientes. Algunos de ellos son:

- Microsoft Windows 95, 98, XP, Vista
- Mac OS X
- Linux
- MS-DOS
- BSDs
- Amiga
- Plataformas que corren la máquina virtual de Java pueden utilizar Jruby.

Al ser open source, existe una gran cantidad de proyectos alrededor de la Web que permiten instalar Ruby de diversas maneras: podemos instalar desde las fuentes (para cuando necesitamos configuraciones específicas) o desde un paquete adecuado a nuestro sistema operativo. La forma más fácil de obtener Ruby es a través de su página web oficial, la cual brinda la seguridad y veracidad del origen de las herramientas y lenguaje de programación.

Figura 15. *En la página oficial de Ruby, encontraremos todas las versiones y las últimas actualizaciones para el lenguaje.*

En nuestro caso, trabajaremos con **Ruby One-Click Installer** sobre un ambiente Windows; aunque los ejemplos y tutoriales funcionan perfectamente sobre otras plataformas, como ser Linux, BSD, MacOS, entre otras. A continuación, veremos cómo instalar Ruby en Windows y en Linux.

Instalar Ruby en Windows

Una vez descargado el paquete One-Click Installer, simplemente lo ejecutamos. Deberemos leer y aceptar su licencia, y, a continuación, elegir las herramientas que instalaremos junto al lenguaje:

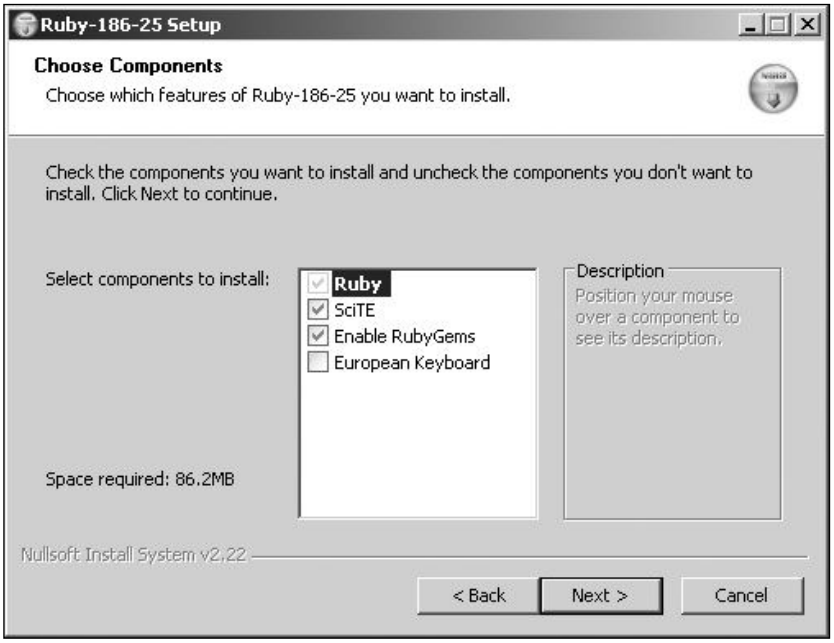

Figura 16. *El paquete nos permite seleccionar herramientas opcionales para instalar. En especial el editor nos será de gran ayuda.*

Luego de la selección de herramientas, definimos la ubicación del directorio de instalación de Ruby y continuamos con ella hasta completarla.

Instalar Ruby en Linux

La instalación de Ruby en Linux presenta dos grandes opciones: podemos bajar el código fuente y compilarlo a mano o, si tenemos alguna distribución con gestores de paquetes, podemos instalarlo directamente desde ellos. Existen paquetes para distintas distribuciones; los más avanzados corresponden a los de Debian o compatibles, aunque si tenemos conocimientos suficientes de Linux, podemos descargar las fuentes y compilar e instalarlo manualmente. Para instalar bajo **Debian** o **Ubuntu**, deberemos tipear en la consola de nuestra distribución Linux:

```
% sudo apt-get install ruby irb rdoc
```

CONOCER EL ENTORNO

Una vez que ha finalizado la instalación en nuestra plataforma, vemos que se han creado una serie de carpetas a modo de jerarquía, en las cuales encontramos el intérprete de Ruby, sus librerías, ejemplos y documentación.

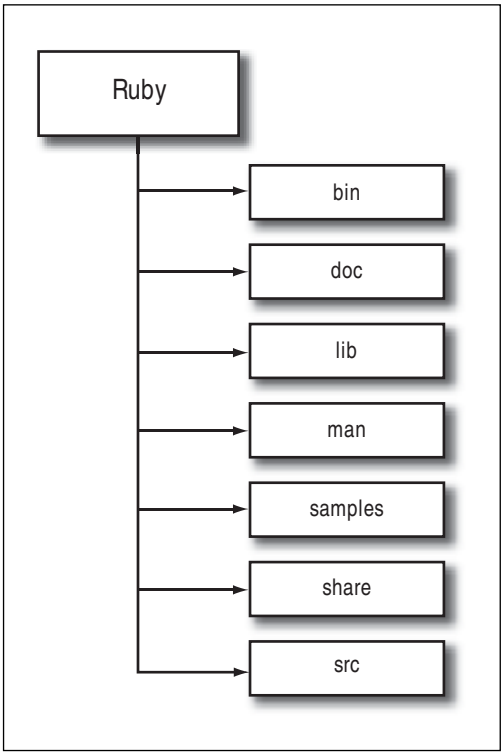

Figura 17. *La estructura de los directorios de Ruby puede resultar familiar a los usuarios de Linux.*

Para empezar a codificar, sólo debemos invocar al intérprete; tenemos dos posibilidades (al menos en Windows). La primera consiste en ejecutar en modo consola, tecleando **irb** desde la línea de comandos.

DOCUMENTACIÓN RUBY

El sitio **www.ruby-lang.org/es/community/mailing-lists/** permite la suscripción a las distintas listas oficiales, una buena forma de mantenernos actualizados. Las listas nos permiten mantener contacto con la comunidad de Ruby y obtener información fiable sobre los desarrollos y proyectos actuales.

Figura 18. *En el intérprete de Ruby, desde la línea de comandos, podemos probar nuestros primeros proyectos.*

Otra opción disponible es ejecutar el **fxri**, que no sólo incorpora al intérprete, sino que posee una ayuda interactiva. Ésta permite que realicemos búsquedas y que conozcamos las principales clases, sus métodos y propiedades.

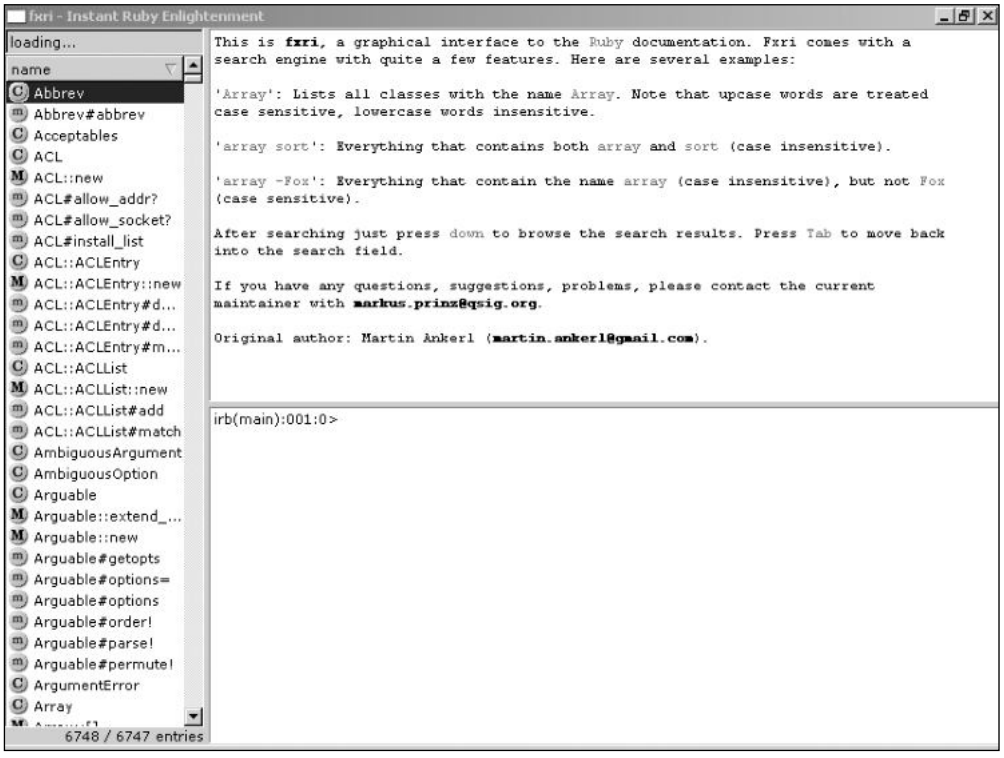

Figura 19. *Entorno simple con ayuda y consola interactiva.*

Recordemos que, al instalar Ruby, tuvimos en dicho proceso la opción de instalar un entorno para desarrollo llamado Scite (**IDE**) que puede ser útil a la hora de realizar nuestras primeras prácticas; sin embargo, lo aconsejable es usar **fxri** y luego saltar al IDE que deseemos o que más nos sea cómodo entre la variedad disponible. Existen muchos entornos con diferentes características; para proyectos más grandes podemos utilizar **Eclipse**, **NetBeans** o **Komodo**.

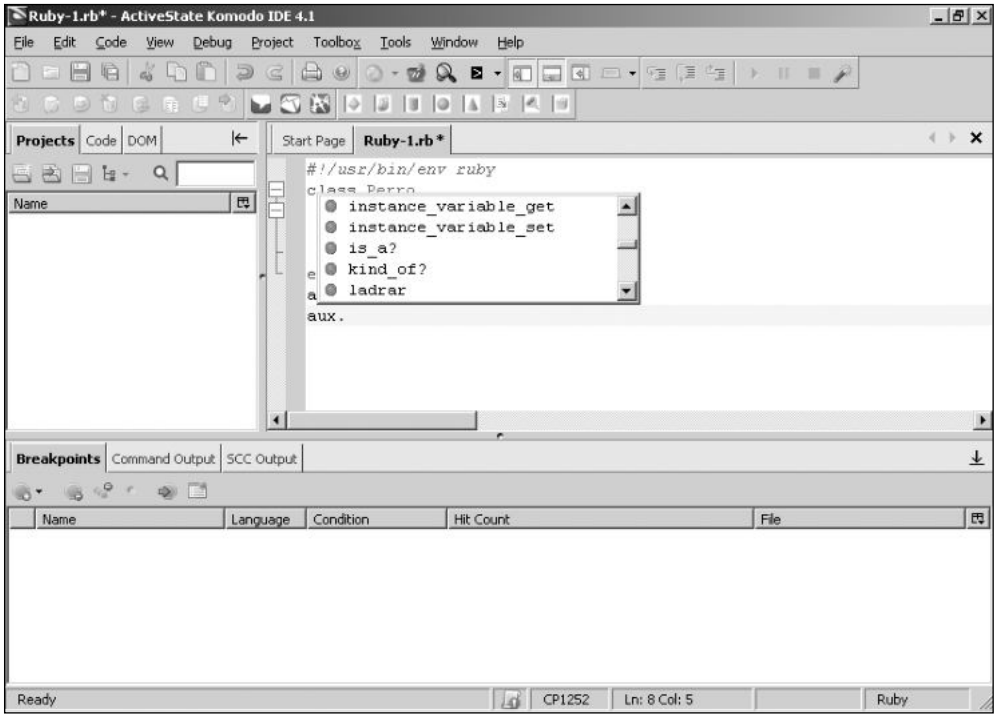

Figura 20. *Komodo es una de las mejores opciones para el desarrollo en Ruby.*

¿Donde obtener ayuda?

Cuando elegimos un lenguaje, debemos observar cuál es la documentación técnica relacionada. En especial, debemos preocuparnos por la calidad en lugar de la cantidad. Éste es un factor importante, porque es preferible que la mayor cantidad de funciones estén documentadas a que se repitan infinitamente en distintos manuales los mismos ejemplos. En el caso de Ruby, podemos elegir entre:

- páginas web;
- IRC;
- foros;
- listas de correo.

Para ponernos en contacto con los sitios de ayuda, es recomendable partir desde la página oficial de acuerdo con nuestros gustos y necesidades.

NUESTRO PRIMER PROGRAMA

Como no podía ser de otra forma, realizaremos el conocido "Hola mundo", pero al estilo Ruby. Abrimos el **irb** o el **fxri** y sólo tipeamos:

```
irb(main):001:0> puts "Hola Mundo"
Hola Mundo
=> nil
```

Ahora hacemos lo mismo, pero agregamos algún operador:

```
irb(main):001:0> puts "Hola Mundo" * 5
Hola MundoHola MundoHola MundoHola MundoHola Mundo
=> nil
```

Vemos que Ruby, nuestro intérprete, diferenciándose de otros lenguajes de programación, es lo suficientemente inteligente como para entender que queremos escribir esa cadena un determinado número de veces.

En este primer capítulo, hemos tenido un acercamiento inicial al lenguaje, conocimos su historia, cómo instalarlo en diferentes entornos y programamos el clásico "Hola mundo". Los próximos capítulos tratan todo lo necesario para convertirnos en expertos.

... RESUMEN

Ruby es un lenguaje de scripts, interpretado, multiplataforma, libre y totalmente orientado a objetos. Su sintaxis simple y su curva de aprendizaje lo sitúan como una alternativa excelente para introducirse en la programación. La libre disponibilidad del lenguaje hace que sea una herramienta para tener en cuenta en entornos empresariales. Ruby permite a los desarrolladores que utilicen términos como elegante, interesante y divertido para describir la experiencia de utilizarlo en el trabajo diario.

TEST DE AUTOEVALUACIÓN

1 Mencione al menos tres características del lenguaje Ruby.

2 ¿Cómo se distribuye Ruby?

3 ¿A qué se denomina Rails?

4 ¿Qué tipo de arquitectura usa Rails?

5 ¿Qué significa que todo es un objeto en Ruby?

6 ¿Para qué sirve el comando irb?

7 ¿Qué se puede hacer con Ruby?

8 ¿Cuáles son las desventajas de que sea interpretado?

9 Mencione algunas deficiencias de Ruby.

10 ¿En qué tareas como desarrollador puede resultarle útil Ruby?