



**ESCUELA SUPERIOR
POLITÉCNICA DEL LITORAL**

Tema: Investigación de Lenguajes

Materia: Lenguajes de Programación

Nombre del grupo en Sidweb: Lenguaje Ruby

Paralelo: 1

Fecha de entrega:
Miércoles, 23 de Octubre del 2013

Lenguaje Ruby

Autores:

- Cristina Barreno
- Sixto Castro
- Jordy Vasquez

23 de octubre de 2013

Índice

1. Introducción	4
2. Características	4
3. Historia	5
4. Tutorial de Instalación	5
5. Hola Mundo y otros Programas Introdutorios	9
6. Referencias	15

Lenguaje Ruby



Figura 1: Logo de Ruby

1. Introducción

Ruby Lenguaje creado por Yukihiro Matsumoto, combina lo mejor de la orientación a objetos (smalltalk) y la facilidad del scripting (perl) generando un lenguaje dinámico, muy expresivo, potente, fácil de aprender y que permite crear aplicaciones empresariales robustas, estables y seguras ademas de ser multiplataforma , es decir puede correr en gran cantidad de arquitecturas , hasta en telefonos celulares, y su implementacion es bajo licencia de software libre .

2. Características

- Ruby es un lenguaje multiplataforma (altamente portable).
- Es fácil de escribir.
- Es un lenguaje interpretado.
- Es un lenguaje orientado a objetos. Al igual que Java, cada tipo de dato es un objeto.
- Es una mezcla de varios lenguajes tales como Perl, Python, Smalltalk y otros, que definen a Ruby como un lenguaje que integra la programación funcional e imperativa.
- Posee expresiones similares(nivel de lenguaje) a las de Perl
- En Ruby, los métodos se pueden o no ser parte de una clase. Puede ser declarado en cualquier parte del archivo.
- Es capaz de manejar excepciones.
- Se puede manejar hilos (multihilos) sin depender del sistema operativo.

3. Historia

Yukihiro Matsumoto crea Ruby en Japon en el año 1993, mientras trabajaba en lenguajes como Perl y PHP, pero se ve tentado en vez de crear un Perl mejorado, su propio lenguaje combinando asi: **Perl, Smaltalk, Eiffel y Lisp**. Su nombre proviene como referencia al lenguaje Perl(perla) el cual originalmente deseaba mejorar.

Se lanza al publico en el año 1995, trayendo asi desarrolladores de todo el mundo, que ven a a Ruby como un lenguaje de exploracion. Tiene un crecimiento lento pero seguro hasta que David Heinemeier Hansson crea el framework Rails , el cual impulsa su crecimiento llevandolo a ser dominado como el **"Lenguaje de Programacion del 2006"** e instalandolo entre los 10 mas populares actualmente.

4. Tutorial de Instalación

Como bien sabemos ruby es un lenguaje multiplataforma por lo que aprenderemos a instalarlo en los diferentes sistemas operativos, asi mismo en todos los casos instalaremos la versión de Ruby **1.9.3** la cual es la version más estable actualmente, ademas te ofrece una extensa lista de paquetes o gemas compatibles y actualizadas.

■ Ruby en Windows

Para instalar el interprete de Ruby en Windows los podemos hacer desde su página oficial visitando <http://rubyinstaller.org/downloads/> donde descargaremos el *RubyInstaller* versión *Ruby1,9,3 – p448* para una arquitectura de 32 bits, si la arquitectura es de 64 bits puedes descargar la versión *Ruby2,0,0 – p247(x64)*.

Después que descarguemos el instalador lo ejecutamos como administrador y empezamos la instalación. Aceptamos el contrato de licencia. Vea *Figura 2*.

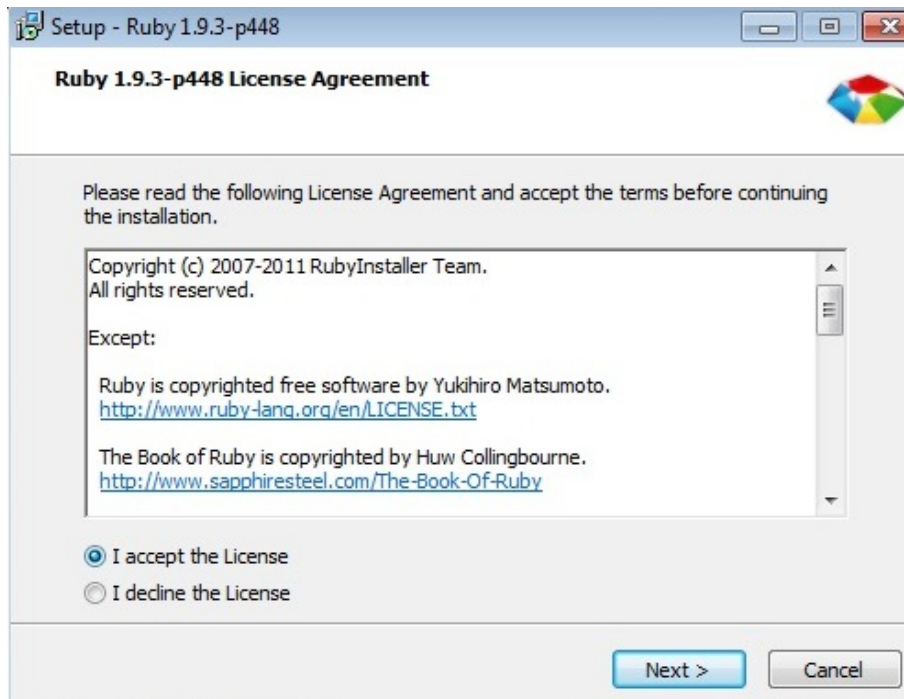


Figura 2: Paso de instalación 1

Agregamos lo necesario para la instalación como se observa en la *Figura3*.

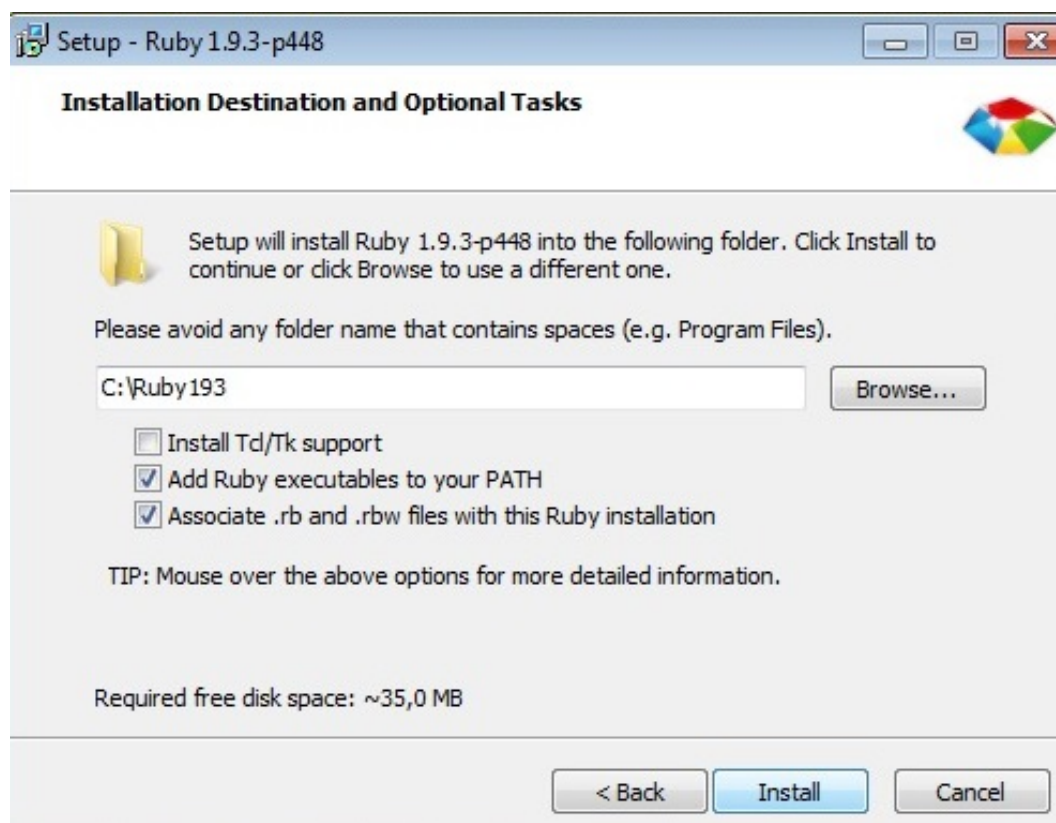


Figura 3: Paso de instalación 2

Terminamos la instalación. Vea *Figura4*.

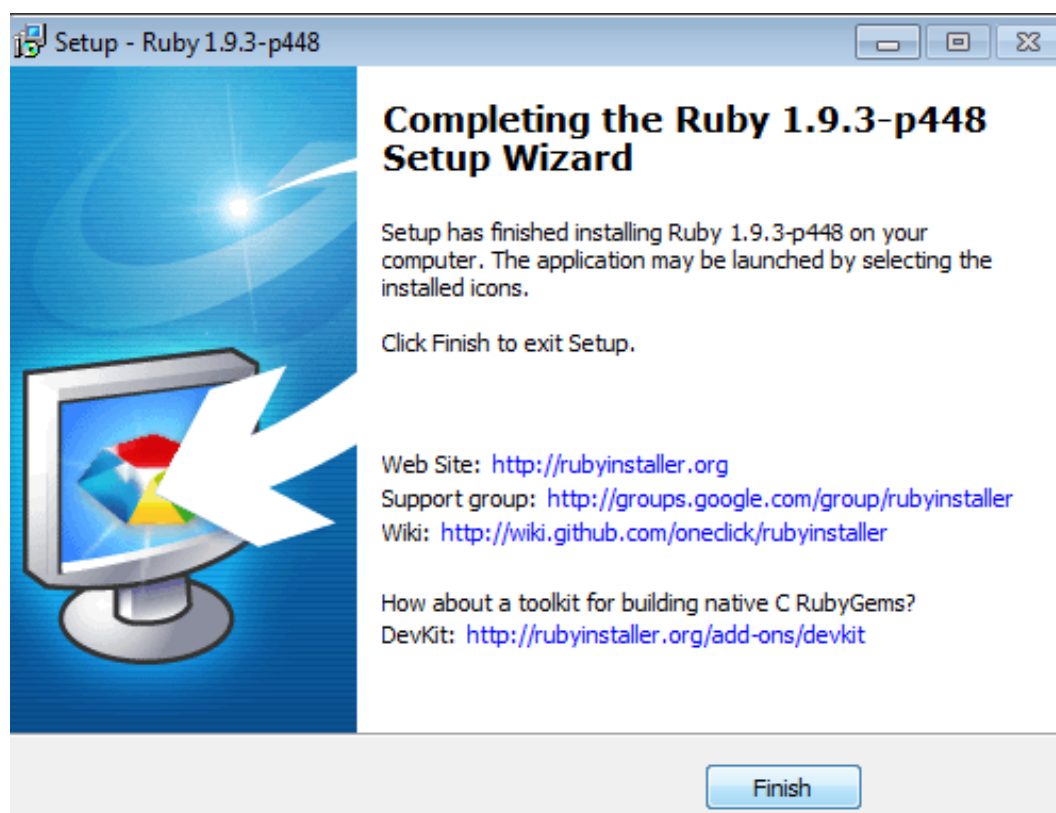


Figura 4: Paso de instalación 3

Abrimos *InteractiveRuby* y comenzamos a trabajar con Ruby desde consola. Vea *Figura5*.



Figura 5: Paso de instalación 4

- **Ruby en Linux** Dependiendo de la distribución que utilices, hay varias maneras de instalar Ruby. En este caso instalaremos la versión Ruby 1.9.3-p125, para esto necesitaremos abrir una terminal, bajaremos el código fuente y lo compilaremos. En la siguiente página web encontraremos un script con los pasos realizados: <http://paste.desdelinux.net/4393>.

5. Hola Mundo y otros Programas Introductorios

A continuación revisaremos algunos programas básicos en ruby para esto solo necesitamos tener instalado en nuestro computador el interprete de ruby, abramos el terminal *InteractiveRuby* y empecemos a programar. Aunque solo trabajaremos desde consola recordemos que esta es una herramienta poderosa con la cual daremos nuestros primeros pasos en el mundo de ruby.

Ejemplo 1: Hola Mundo

Para hacer que nuestro programa escriba "*HolaMundo*" usaremos el comando *putsHolaMundo*", siendo "*puts*" el comando básico para escribir algo en ruby.

```
puts "Hola Mundo"
```

Programa Hola Mundo

Ejemplo 2: Trabajando con arreglos y Strings

- Crear un array

```
@names = Array.new
```

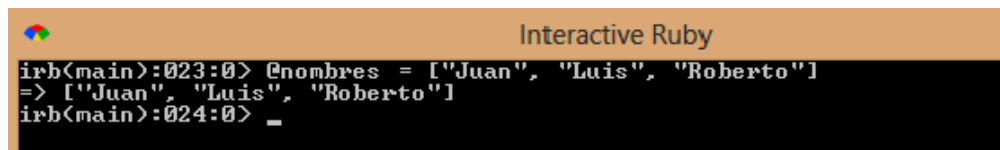
Crea un Array vacío.

- Método []

```
@names = ["Juan", "Luis", "Roberto"]
```

Crea el Array y lo inicializa con strings.

Salida



```
irb(main):023:0> @nombres = ["Juan", "Luis", "Roberto"]  
=> ["Juan", "Luis", "Roberto"]  
irb(main):024:0> _
```

- Método: +

```
@clientes = ["Juan", "Luis", "Roberto"]
@clientes2 = ["Pierre", "Joe", "Rachel"]
@clientes + @clientes2
```

Método para unir un array al final de otro.

Salida

```
Interactive Ruby
irb(main):003:0> @nombres = ["Juan", "Luis", "Roberto"]
=> ["Juan", "Luis", "Roberto"]
irb(main):004:0> @nombres2 = ["Pierre", "Joe", "Rachel"]
=> ["Pierre", "Joe", "Rachel"]
irb(main):005:0> @nombres + @nombres2
=> ["Juan", "Luis", "Roberto", "Pierre", "Joe", "Rachel"]
```

- Ordenar un arreglo

```
notasdeberes = [2,4,3,6,8,9,5,1]
notasdeberes.sort
```

Ordena un Array de menor a mayor.

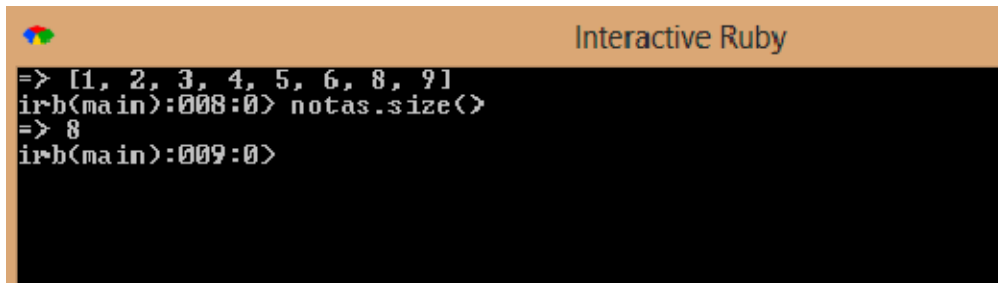
Salida

```
Interactive Ruby
irb(main):006:0> notas=[2,4,3,6,8,9,5,1]
=> [2, 4, 3, 6, 8, 9, 5, 1]
irb(main):007:0> notas.sort
=> [1, 2, 3, 4, 5, 6, 8, 9]
```

- Obtener el tamaño de un arreglo

```
notasdeberes.size()
```

Salida

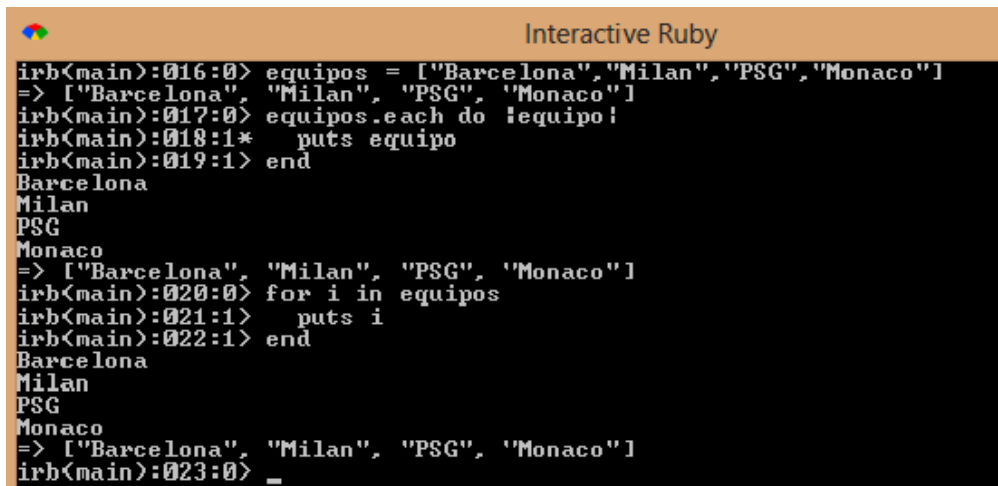


```
Interactive Ruby
=> [1, 2, 3, 4, 5, 6, 8, 9]
irb(main):008:0> notas.size<>
=> 8
irb(main):009:0>
```

- Recorrer un arreglo: each

```
equipos.each do |equipo|
  puts equipo
end
```

Salida



```
Interactive Ruby
irb(main):016:0> equipos = ["Barcelona", "Milan", "PSG", "Monaco"]
=> ["Barcelona", "Milan", "PSG", "Monaco"]
irb(main):017:0> equipos.each do |equipo|
irb(main):018:1*   puts equipo
irb(main):019:1> end
Barcelona
Milan
PSG
Monaco
=> ["Barcelona", "Milan", "PSG", "Monaco"]
irb(main):020:0> for i in equipos
irb(main):021:1>   puts i
irb(main):022:1> end
Barcelona
Milan
PSG
Monaco
=> ["Barcelona", "Milan", "PSG", "Monaco"]
irb(main):023:0> _
```

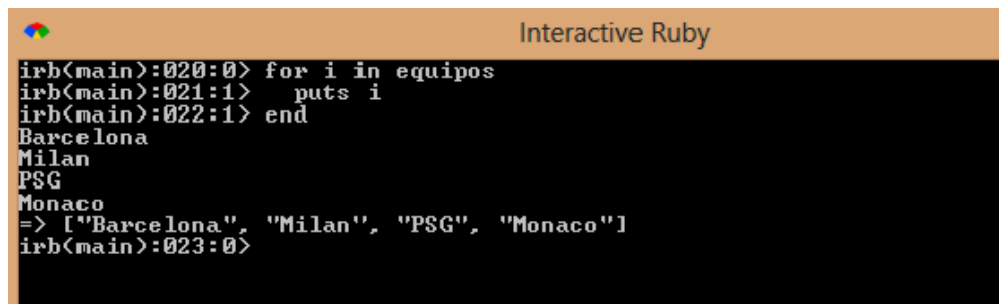
El método 'each' lo que va hacer es recorrer cada elemento del arreglo equipos. En cada iteración, cada elemento del array se guarda en

la variable equipo y se va imprimir cada elemento(cadena) del arreglo equipos ya que se encuentra la función puts que se encarga de imprimir cada cadena.

- Recorrer el arreglo: for in

```
for i in equipos
  puts i
end
```

Salida



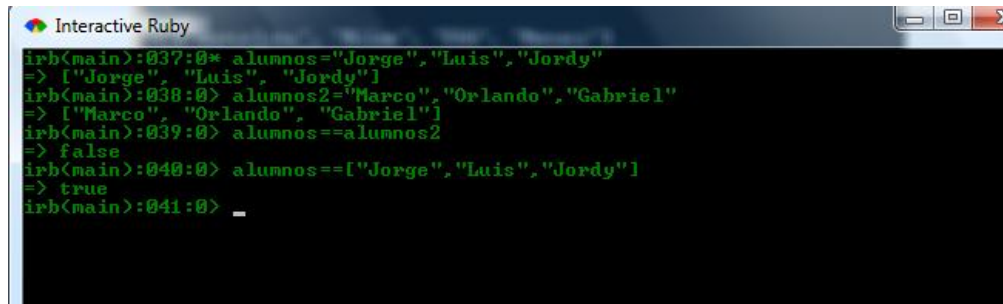
```
Interactive Ruby
irb(main):020:0> for i in equipos
irb(main):021:1>   puts i
irb(main):022:1> end
Barcelona
Milan
PSG
Monaco
=> ["Barcelona", "Milan", "PSG", "Monaco"]
irb(main):023:0>
```

Este caso es parecido al ejercicio anterior, sino que ahora se utiliza un ciclo 'for' para recorrer el arreglo. En cada iteración, la variable 'i' toma el valor actual del array equipos, y luego va imprimir dicho elemento ya que se hace un put en cada iteración.

- Igualdad entre arreglos

```
alumnos1=[" Jorge "," Luis "," Jordy "]  
alumnos2=[" Orlando "," Gabriel "," Steven "]  
alumnos1==alumnos2
```

Salida



```
Interactive Ruby  
irb(main):037:0* alumnos="Jorge","Luis","Jordy"  
=> ["Jorge", "Luis", "Jordy"]  
irb(main):038:0> alumnos2="Marco","Orlando","Gabriel"  
=> ["Marco", "Orlando", "Gabriel"]  
irb(main):039:0> alumnos==alumnos2  
=> false  
irb(main):040:0> alumnos=["Jorge","Luis","Jordy"]  
=> true  
irb(main):041:0> _
```

Devuelve false si no son iguales y true si lo son.

Ejemplo 3: Objetos y métodos

- Ejemplo de crear un objeto y un método

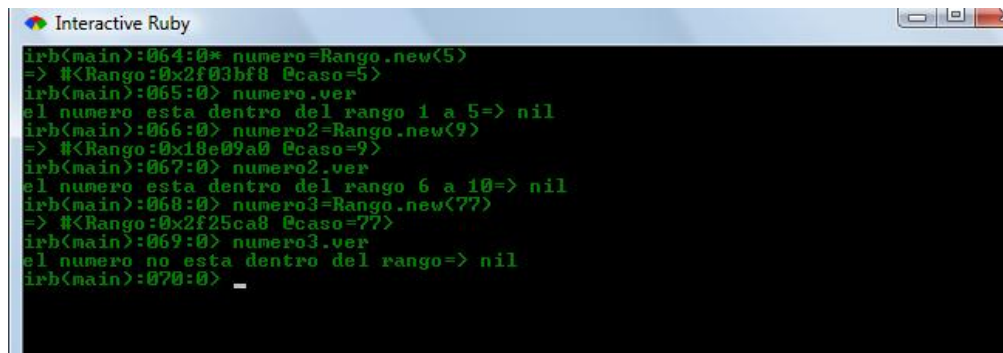
```
class Rango  
  def initialize(caso=0)  
    @caso=caso  
  end  
  def ver  
    case @caso  
    when 1, 2..5  
      print"el numero esta dentro del rango 1 a 5"  
    when 6.. 10  
      print"el numero esta dentro del rango 6 a 10"  
    else  
      print"el numero esta fuera del rango"  
    end  
  end  
end
```

Las palabras claves son class para darle el nombre a nuestro objeto, def initialize para inicializarlo , y def ...(y el nombre del metodo) para crear un metodo para el objeto

Utilizando el objeto y su método

```
numero=Rango.new(5)
numero.ver
```

Salida



```
Interactive Ruby
irb(main):064:0* numero=Rango.new(5)
=> #<Rango:0x2f03bf8 @caso=5>
irb(main):065:0> numero.ver
el numero esta dentro del rango 1 a 5=> nil
irb(main):066:0> numero2=Rango.new(9)
=> #<Rango:0x18e09a0 @caso=9>
irb(main):067:0> numero2.ver
el numero esta dentro del rango 6 a 10=> nil
irb(main):068:0> numero3=Rango.new(77)
=> #<Rango:0x2f25ca8 @caso=77>
irb(main):069:0> numero3.ver
el numero no esta dentro del rango=> nil
irb(main):070:0> _
```

Salida

Hacemos una instancia del objeto poniendo el [nombre del objeto].new, e inicializandolo ya sea con vacío o con un valor, y para utilizar el metodo de la misma forma [nombre del objeto].[nombe del método] como muestra Numero2.ver.

6. Referencias

1. <https://www.ruby-lang.org/es/about/>
2. http://es.wikibooks.org/wiki/Programaci3n_en_Ruby/Historia
3. <http://www.maestrosdelweb.com/editorial/la-clase-array-en-ruby/>
4. <https://www.ruby-lang.org/es/documentation/quickstart/2/>
5. <http://eudev2.uta.cl/rid=1GR0DSG4D-1Y1NH87-4RQ/ruby.pdf>
6. http://www.ecured.cu/index.php/Lenguaje_de_Programaci3n_Ruby
7. <http://es.wikipedia.org/wiki/Ruby#Caracter.C3.ADsticas>
8. http://es.wikibooks.org/wiki/Programaci3n_en_Ruby#Caracter.C3.ADsticas_Especiales_del_Lenguaje
9. http://www.ecured.cu/index.php/Lenguaje_de_Programaci3n_Ruby#Caracter.C3.ADsticas_generales_del_lenguaje
10. <https://www.ruby-lang.org/es/downloads/>