Ruby

por Giménez Silva Germán Alberto

Facultad de Ciencia y Tecnologia de la Uader
Proyecto de migración a Gnu/Linux de la Universidad Autónoma de Entre Ríos
Silix - Soluciones Libres
Lug Oro Verde

August 27, 2007

Historia de Ruby:

Ruby

Ruby es un lenguaje de programación reflexivo y orientado a objetos creado por el programador japonés Yukihiro "Matz" Matsumoto en 1993. Combina una sintaxis inspirada en Python, Perl con características de programación orientada a objetos similares a Smalltalk. Comparte también funcionalidad con otros lenguajes de programación como Lisp, Lua, Dylan y CLU. Ruby es un lenguaje de programación interpretado en una sola pasada y su implementación oficial es distribuida bajo una licencia de software libre.

Licencia:

Licencia

El intérprete y las bibliotecas están licenciadas de forma dual (inseparable) bajo las licencias libres y de código abierto GPL y Licencia Ruby.

Sobre lo que no trata esta charla:

Sobre lo que no trata esta charla:

- Patrón Modelo Vista Controlador

Sobre lo que no trata esta charla:

- Patrón Modelo Vista Controlador
- Ruby on Rails

Sobre lo que no trata esta charla:

- Patrón Modelo Vista Controlador
- Ruby on Rails
- Scaffolding

Propaganda

Si alguien quiere ver eso debe ir a la charla que sigue, que seguramente estará muy buena. Yo al menos no me la voy a perder.

Sobre lo que si trata esta charla:

Sobre lo que si trata esta charla:

- Introducción a la programación con Ruby

Sobre lo que si trata esta charla:

- Introducción a la programación con Ruby
- Algunos conceptos básicos

Sobre lo que si trata esta charla:

- Introducción a la programación con Ruby
- Algunos conceptos básicos
- Algunos Scripts interesantes

Sobre lo que si trata esta charla:

- Introducción a la programación con Ruby
- Algunos conceptos básicos
- Algunos Scripts interesantes
- Algunas librerías

Objetivo

El fin de esta charla es solamente despertar el interés de quienes asistan a la misma por el lenguaje Ruby, un lenguaje innovador y con muchas características de última generación.

1º Programa:

Hola Mundo

Todo empieza con un "hola mundo".

```
#!/usr/bin/env ruby
puts "Hola Mundo"
```

1° Programa:

Hola Mundo

Todo empieza con un "hola mundo".

```
#!/usr/bin/env ruby
puts "Hola Mundo"

# 1° Forma de ejecutar el programa
# chmod 755 ejemplo0-1.rb
# ./ejemplo0-1.rb
```

1° Programa:

Hola Mundo

Todo empieza con un "hola mundo".

```
#!/usr/bin/env ruby
puts "Hola Mundo"

# 1° Forma de ejecutar el programa
# chmod 755 ejemplo0-1.rb
# ./ejemplo0-1.rb

# 2° Forma de ejecutar el programa
# ruby ejemplo0-1.rb
```

Instalando ruby en Debian:

Instalando en Debian

```
ggerman@simplondio:~$ su
Password: *********
simplondio:/home/ggerman#apt-get install ruby
simplondio:/home/ggerman#apt-get install libdbd-mysql-ruby
simplondio:/home/ggerman#apt-get install libgd-ruby1.8
```

Instalando ruby en Debian:

Instalando en Debian

```
ggerman@simplondio:~$ su
Password: *********
simplondio:/home/ggerman#apt-get install ruby
simplondio:/home/ggerman#apt-get install libdbd-mysql-ruby
simplondio:/home/ggerman#apt-get install libgd-ruby1.8
```

Nota la documentación de cada paquete se instala en:

Ruby	/usr/share/doc/ruby/
Ruby Mysql	/usr/share/doc/libmysql-ruby1.8/
Ruby Gd	/usr/share/doc/libgd-ruby1.8/

Palabras reservadas:

Palabras reservadas

BEGIN	END	alias	and	begin
break	case	class	def	defined?
do	else	elsif	end	ensure
false	for	if	in	module
next	nil	not	or	redo
rescue	retry	return	self	super
then	true	undef	unless	until
when	while	yield		

Variables y estructuras de datos:

Tipos y estructuras de datos

Algunos de los tipo de datos y estructuras que nos ofrece Ruby. Como mencionamos previamente todos estos tipos son objetos en si mismos, y tienen sus propios métodos.

Variables y estructuras de datos:

Tipos y estructuras de datos

Algunos de los tipo de datos y estructuras que nos ofrece Ruby. Como mencionamos previamente todos estos tipos son objetos en si mismos, y tienen sus propios métodos.

Sentencia	Tipo
a = "Hola"	String
nro = 5.5	Float
nro = 5	Fixnum
b = Array[1, 2, 3]	Array
c1 = {"*flat*" => 3, "*curved*" => 2}	Hashes

Tipos de datos:

Conociendo el tipo de dato:

```
#!/usr/bin/env ruby
print "Hola".class
print 20.class
print 252431930000000.class
print [1,2,3].class
print ("a" .. "z").class

ggerman@simplondio:/examples$ ruby ejemplo09.rb
String
Fixnum
Bignum
Array
Range
```

Ruby interprete linea a linea:

Interactivo

Al igual que python ruby incluye una aplicación para programar de manera interactiva y así poder depurar nuestros programas.
Para hacer uso de este script en Debian debemos instalar el paquete ruby1.8-examples (o 1.9 dependiendo la versión que estemos utilizando).

```
$ ruby /usr/share/doc/ruby1.8-examples/examples/eval.rb
ruby> 5.times{print "Linea a Linea \n"}
Linea a Linea
Linea a Linea
Linea a Linea
Linea a Linea
Linea a Linea
5
ruby> exit
ggerman@simplondio:~$
```

apt-get install ruby1.8-examples

Modificando un objeto:

Creando una clase virtual

Ruby nos permite crear clases virtuales para instancias de objetos específicos. Definiendo las propiedades (métodos y constantes) de la clase.

Modificando un objeto:

Creando una clase virtual

Ruby nos permite crear clases virtuales para instancias de objetos específicos. Definiendo las propiedades (métodos y constantes) de la clase.

```
#!/usr/bin/env ruby
variable = "80"
class << variable
        def cien_porciento
                (self.to_f / 100).to_f
        end
end
print variable.cien_porciento
print "\n"
ggerman@simplondio:~/examples$ ruby inherited.rb
0.8
```

Modificando un objeto:

Creando una clase virtual

Ruby nos permite crear clases virtuales para instancias de objetos específicos. Definiendo las propiedades (métodos y constantes) de la clase.

```
#!/usr/bin/env ruby
variable = "80"
class << variable
        def cien_porciento
                (self.to_f / 100).to_f
        end
end
print variable.cien_porciento
print "\n"
ggerman@simplondio:~/examples$ ruby inherited.rb
0.8
```

Como vemos en el código creamos el método **cien_porciento** para el objeto variable de contenido 80.

Funciones:

Creando función

Ahora veremos como utilizar expresiones regulares en la búsqueda de una cadena de texto.

Funciones:

Creando función

Ahora veremos como utilizar expresiones regulares en la búsqueda de una cadena de texto.

```
#!/usr/bin/env ruby

def buscar(cadena)
        cadena =~ /mundo/
end

print buscar("Hola mundo")
print "\n"
ggerman@simplondio:~/examples$ ruby ejemplo07.rb
5
```

Funciones:

Creando función

Ahora veremos como utilizar expresiones regulares en la búsqueda de una cadena de texto.

```
#!/usr/bin/env ruby

def buscar(cadena)
        cadena =~ /mundo/
end

print buscar("Hola mundo")
print "\n"
ggerman@simplondio:~/examples$ ruby ejemplo07.rb
5
```

Sencillo ¿no?

Utilizando bloques y llamando al sistema

Listando Archivos v1.0

```
#!/usr/bin/env ruby
lista = `ls -l -h`
lista.each do |regLista|
  linea = regLista.split
  print "Usuario: #{linea[3]}"
  print " Tamaño: #{linea[4]}"
  print " Nombre: #{linea[7]}"
  print "\n"
end
```

Utilizando bloques y llamando al sistema

Listando Archivos v1.0

```
#!/usr/bin/env ruby
lista = `ls -l -h`
lista.each do |regLista|
  linea = regLista.split
  print "Usuario: #{linea[3]}"
  print " Tamaño: #{linea[4]}"
  print " Nombre: #{linea[7]}"
  print "\n"
end
```

Salida

```
ggerman@simplondio:~/software$ ruby ejemplo1.rb
Usuario:  ggerman Tamaño:  650 Nombre: archivo02.txt
Usuario:  ggerman Tamaño:  50  Nombre: archivo.txt
....
```

Utilizando bloques y llamando al sistema

Listando Archivos v1.1

```
#!/usr/bin/env ruby
lista = `ls -l -h`
lista.each {|x| print "Usuario:  #{x.split[3]}
                       Tamaño:  #{x.split[4]}
                       Nombre: #{x.split[7]} \n"}
```

Utilizando bloques y llamando al sistema

Listando Archivos v1.1

```
#!/usr/bin/env ruby
lista = `ls -l -h`
lista.each {|x| print "Usuario:  #{x.split[3]}
                       Tamaño:  #{x.split[4]}
                       Nombre: #{x.split[7]} \n"}
```

Salida

```
ggerman@simplondio:~/software$ ruby ejemplo1.rb
Usuario:  ggerman Tamaño:  650 Nombre: archivo02.txt
Usuario:  ggerman Tamaño:  50  Nombre: archivo.txt
Usuario:  ggerman Tamaño:  548 Nombre: basic01.rb
Usuario:  ggerman Tamaño:  79  Nombre: basic01.rb~
....
```

Trabajando con Strings:

Acciones con String

Sentencia	Salida	Acción
a = "Hola Mundo"		Asignación
print a	"Hola Mundo"	Ecritura
print a.length	10	Largo de la cadena
print "Hola Mundo".uppcase	HOLA MUNDO	Método uppcase
print "Hola Mundo".center(80)	Hola Mundo	Texto centrado
a.each_byte{|char| print char,""}	72 111 108 97 ...	
a.split	['Hola', 'Mundo']	
print a.center(16,'#')	*###HolaMundo###*	Centra el texto entre #
a.swapcase	hOLA mUNDO	
print a.index("undo")	6	Busca la cadena "undo"

Trabajando con Strings:

Recorriendo un listado de String

```
("a".."z").each { |x| print "#{x}, " }
```

Trabajando con Strings:

Recorriendo un listado de String

```
("a".."z").each { |x| print "#{x}, " }
```

Salida

```
ggerman@simplondio:~/software$ ruby -d ejemplo2.rb
a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s,
t, u, v, w, x, y, z,
ggerman@simplondio:~/software$
```

Librerías:

Integración de Ruby con librerías externas

Ruby es un lenguaje que se integra con una gran cantidad de librerías, por ejemplo:

Librerías:

Integración de Ruby con librerías externas

Ruby es un lenguaje que se integra con una gran cantidad de librerías, por ejemplo:

Librerias gráficas

Qt, Gtk, Tk, Ncurses, etc...

Librerías:

Integración de Ruby con librerías externas

Ruby es un lenguaje que se integra con una gran cantidad de librerías, por ejemplo:

Librerias gráficas

Qt, Gtk, Tk, Ncurses, etc...

Nota: Ruby también trabaja con librerías privativas tales como la Api de Win. Pero como no utilizo este tipo de librerías ni el SO Win, no recomiendo utilizarlos ya que la aplicación que desarrollen integrando estas herramientas quedará a merced de licencias restrictivas y privativas.
De todas maneras pueden encontrar material sobre el tema en **Ruby Developer's Guide** pag. 111, 482

Librerias:

Librerías Bases de Datos

Ruby/DBI consiste en multiples drivers para conectarse a bases de datos. **Ruby Developer's Guide** Cap 3 Mysql trae una libreria para integrarse directamente con Ruby y es la que veremos.

Librerias:

Librerias de tratamiento de imágenes

OpenGl, GD, etc ...

Librerias:

Librerias de tratamiento de imágenes

OpenGl, GD, etc ...

Bindings

Java, Python, etc ...

mysql:

Mysql - Trabajando con Base de Datos

Una de las bases de datos que podemos utilizar con Ruby es Mysql. Esta base de datos esta muy difundida dentro de las aplicaciones que podemos desarrollar con software libre.

```
#/usr/bin/env ruby!
```

mysql:

Mysql - Trabajando con Base de Datos

Una de las bases de datos que podemos utilizar con Ruby es Mysql. Esta base de datos esta muy difundida dentro de las aplicaciones que podemos desarrollar con software libre.

```
#/usr/bin/env ruby!
require "mysql" # Incluyo en la aplicación la libmysql
```

mysql:

Mysql - Trabajando con Base de Datos

Una de las bases de datos que podemos utilizar con Ruby es Mysql. Esta base de datos esta muy difundida dentro de las aplicaciones que podemos desarrollar con software libre.

```
#/usr/bin/env ruby!
require "mysql" # Incluyo en la aplicación la libmysql
connect = Mysql::new("localhost", "root") # Conexión al motor de base de datos mysql
```

mysql:

Mysql - Trabajando con Base de Datos

Una de las bases de datos que podemos utilizar con Ruby es Mysql. Esta base de datos esta muy difundida dentro de las aplicaciones que podemos desarrollar con software libre.

```
#/usr/bin/env ruby!
require "mysql" # Incluyo en la aplicación la libmysql
connect = Mysql::new("localhost", "root") # Conexión al motor de base de datos mysql
sql = "SHOW DATABASES;" # Sentencia sql que devuelve el listado de Bases de Datos
```

mysql:

Mysql - Trabajando con Base de Datos

Una de las bases de datos que podemos utilizar con Ruby es Mysql. Esta base de datos esta muy difundida dentro de las aplicaciones que podemos desarrollar con software libre.

```
#/usr/bin/env ruby!
require "mysql" # Incluyo en la aplicación la libmysql
connect = Mysql::new("localhost", "root") # Conexión al motor de base de datos mysql
sql = "SHOW DATABASES;" # Sentencia sql que devuelve el listado de Bases de Datos
result = connect.query(sql) # Ejecución del Sql y almacenamiento del resultado
```

mysql:

Mysql - Trabajando con Base de Datos

Una de las bases de datos que podemos utilizar con Ruby es Mysql. Esta base de datos esta muy difundida dentro de las aplicaciones que podemos desarrollar con software libre.

```
#/usr/bin/env ruby!
require "mysql" # Incluyo en la aplicación la libmysql
connect = Mysql::new("localhost", "root") # Conexión al motor de base de datos mysql
sql = "SHOW DATABASES;" # Sentencia sql que devuelve el listado de Bases de Datos
result = connect.query(sql) # Ejecución del Sql y almacenamiento del resultado
result.each do |regField| # Recorre el listado de registros arrojados
```

mysql:

Mysql - Trabajando con Base de Datos

Una de las bases de datos que podemos utilizar con Ruby es Mysql. Esta base de datos esta muy difundida dentro de las aplicaciones que podemos desarrollar con software libre.

```
#/usr/bin/env ruby!
require "mysql" # Incluyo en la aplicación la libmysql
connect = Mysql::new("localhost", "root") # Conexión al motor de base de datos mysql
sql = "SHOW DATABASES;" # Sentencia sql que devuelve el listado de Bases de Datos
result = connect.query(sql) # Ejecución del Sql y almacenamiento del resultado
result.each do |regField| # Recorre el listado de registros arrojados
    nombre = regField[0] # Almacena el nombre de las Bases de Datos
```

mysql:

Mysql - Trabajando con Base de Datos

Una de las bases de datos que podemos utilizar con Ruby es Mysql. Esta base de datos esta muy difundida dentro de las aplicaciones que podemos desarrollar con software libre.

```
#/usr/bin/env ruby!
require "mysql" # Incluyo en la aplicación la libmysql
connect = Mysql::new("localhost", "root") # Conexión al motor de base de datos mysql
sql = "SHOW DATABASES;" # Sentencia sql que devuelve el listado de Bases de Datos
result = connect.query(sql) # Ejecución del Sql y almacenamiento del resultado
result.each do |regField| # Recorre el listado de registros arrojados
    nombre = regField[0] # Almacena el nombre de las Bases de Datos
    print "* #{nombre} \n" # Muestro por la salida estandar el nombre de la BD
```

mysql:

Mysql - Trabajando con Base de Datos

Una de las bases de datos que podemos utilizar con Ruby es Mysql. Esta base de datos esta muy difundida dentro de las aplicaciones que podemos desarrollar con software libre.

```
#/usr/bin/env ruby!
require "mysql" # Incluyo en la aplicación la libmysql
connect = Mysql::new("localhost", "root") # Conexión al motor de base de datos mysql
sql = "SHOW DATABASES;" # Sentencia sql que devuelve el listado de Bases de Datos
result = connect.query(sql) # Ejecución del Sql y almacenamiento del resultado
result.each do |regField| # Recorre el listado de registros arrojados
    nombre = regField[0] # Almacena el nombre de las Bases de Datos
    print "* #{nombre} \n" # Muestro por la salida estandar el nombre de la BD
    database = Mysql::new("localhost", "root",nil , nombre); # Me conecto a la BD
```

mysql:

Mysql - Trabajando con Base de Datos

Una de las bases de datos que podemos utilizar con Ruby es Mysql. Esta base de datos esta muy difundida dentro de las aplicaciones que podemos desarrollar con software libre.

```
#/usr/bin/env ruby!
require "mysql" # Incluyo en la aplicación la libmysql
connect = Mysql::new("localhost", "root") # Conexión al motor de base de datos mysql
sql = "SHOW DATABASES;" # Sentencia sql que devuelve el listado de Bases de Datos
result = connect.query(sql) # Ejecución del Sql y almacenamiento del resultado
result.each do |regField| # Recorre el listado de registros arrojados
    nombre = regField[0] # Almacena el nombre de las Bases de Datos
    print "* #{nombre} \n" # Muestro por la salida estandar el nombre de la BD
    database = Mysql::new("localhost", "root",nil , nombre); # Me conecto a la BD
    sql = "SHOW TABLES"; # Sql que consulta las tablas dentro de la BD
```

mysql:

Mysql - Trabajando con Base de Datos

Una de las bases de datos que podemos utilizar con Ruby es Mysql. Esta base de datos esta muy difundida dentro de las aplicaciones que podemos desarrollar con software libre.

```
#/usr/bin/env ruby!
require "mysql" # Incluyo en la aplicación la libmysql
connect = Mysql::new("localhost", "root") # Conexión al motor de base de datos mysql
sql = "SHOW DATABASES;" # Sentencia sql que devuelve el listado de Bases de Datos
result = connect.query(sql) # Ejecución del Sql y almacenamiento del resultado
result.each do |regField| # Recorre el listado de registros arrojados
    nombre = regField[0] # Almacena el nombre de las Bases de Datos
    print "* #{nombre} \n" # Muestro por la salida estandar el nombre de la BD
    database = Mysql::new("localhost", "root",nil , nombre); # Me conecto a la BD
    sql = "SHOW TABLES"; # Sql que consulta las tablas dentro de la BD
    tables = database.query(sql) # Ejecución del Sql y almacenamiento del resultado
```

mysql:

Mysql - Trabajando con Base de Datos

Una de las bases de datos que podemos utilizar con Ruby es Mysql. Esta base de datos esta muy difundida dentro de las aplicaciones que podemos desarrollar con software libre.

```
#/usr/bin/env ruby!
require "mysql" # Incluyo en la aplicación la libmysql
connect = Mysql::new("localhost", "root") # Conexión al motor de base de datos mysql
sql = "SHOW DATABASES;" # Sentencia sql que devuelve el listado de Bases de Datos
result = connect.query(sql) # Ejecución del Sql y almacenamiento del resultado
result.each do |regField| # Recorre el listado de registros arrojados
    nombre = regField[0] # Almacena el nombre de las Bases de Datos
    print "* #{nombre} \n" # Muestro por la salida estandar el nombre de la BD
    database = Mysql::new("localhost", "root",nil , nombre); # Me conecto a la BD
    sql = "SHOW TABLES"; # Sql que consulta las tablas dentro de la BD
    tables = database.query(sql) # Ejecución del Sql y almacenamiento del resultado
    tables.each do |regTables| # Recorre el listado de Tablas
```

mysql:

Mysql - Trabajando con Base de Datos

Una de las bases de datos que podemos utilizar con Ruby es Mysql. Esta base de datos esta muy difundida dentro de las aplicaciones que podemos desarrollar con software libre.

```
#/usr/bin/env ruby!
require "mysql" # Incluyo en la aplicación la libmysql
connect = Mysql::new("localhost", "root") # Conexión al motor de base de datos mysql
sql = "SHOW DATABASES;" # Sentencia sql que devuelve el listado de Bases de Datos
result = connect.query(sql) # Ejecución del Sql y almacenamiento del resultado
result.each do |regField| # Recorre el listado de registros arrojados
    nombre = regField[0] # Almacena el nombre de las Bases de Datos
    print "* #{nombre} \n" # Muestro por la salida estandar el nombre de la BD
    database = Mysql::new("localhost", "root",nil , nombre); # Me conecto a la BD
    sql = "SHOW TABLES"; # Sql que consulta las tablas dentro de la BD
    tables = database.query(sql) # Ejecución del Sql y almacenamiento del resultado
    tables.each do |regTables| # Recorre el listado de Tablas
        print " - #{regTables[0]}" # Escribo el nombre de la tabla Uno a uno
```

mysql:

Mysql - Trabajando con Base de Datos

Una de las bases de datos que podemos utilizar con Ruby es Mysql. Esta base de datos esta muy difundida dentro de las aplicaciones que podemos desarrollar con software libre.

```
#/usr/bin/env ruby!
require "mysql" # Incluyo en la aplicación la libmysql
connect = Mysql::new("localhost", "root") # Conexión al motor de base de datos mysql
sql = "SHOW DATABASES;" # Sentencia sql que devuelve el listado de Bases de Datos
result = connect.query(sql) # Ejecución del Sql y almacenamiento del resultado
result.each do |regField| # Recorre el listado de registros arrojados
    nombre = regField[0] # Almacena el nombre de las Bases de Datos
    print "* #{nombre} \n" # Muestro por la salida estandar el nombre de la BD
    database = Mysql::new("localhost", "root",nil , nombre); # Me conecto a la BD
    sql = "SHOW TABLES"; # Sql que consulta las tablas dentro de la BD
    tables = database.query(sql) # Ejecución del Sql y almacenamiento del resultado
    tables.each do |regTables| # Recorre el listado de Tablas
        print " - #{regTables[0]}" # Escribo el nombre de la tabla Uno a uno
        sql = "SELECT * FROM #{regTables[0]};" # Sql que consulta todos los registros de la tabla
```

mysql:

Mysql - Trabajando con Base de Datos

Una de las bases de datos que podemos utilizar con Ruby es Mysql. Esta base de datos esta muy difundida dentro de las aplicaciones que podemos desarrollar con software libre.

```
#/usr/bin/env ruby!
require "mysql" # Incluyo en la aplicación la libmysql
connect = Mysql::new("localhost", "root") # Conexión al motor de base de datos mysql
sql = "SHOW DATABASES;" # Sentencia sql que devuelve el listado de Bases de Datos
result = connect.query(sql) # Ejecución del Sql y almacenamiento del resultado
result.each do |regField| # Recorre el listado de registros arrojados
    nombre = regField[0] # Almacena el nombre de las Bases de Datos
    print "* #{nombre} \n" # Muestro por la salida estandar el nombre de la BD
    database = Mysql::new("localhost", "root",nil , nombre); # Me conecto a la BD
    sql = "SHOW TABLES"; # Sql que consulta las tablas dentro de la BD
    tables = database.query(sql) # Ejecución del Sql y almacenamiento del resultado
    tables.each do |regTables| # Recorre el listado de Tablas
        print " - #{regTables[0]}" # Escribo el nombre de la tabla Uno a uno
        sql = "SELECT * FROM #{regTables[0]};" # Sql que consulta todos los registros de la tabla
        count = database.query(sql) # Ejecución del Sql previo y almacenamiento
```

mysql:

Mysql - Trabajando con Base de Datos

Una de las bases de datos que podemos utilizar con Ruby es Mysql. Esta base de datos esta muy difundida dentro de las aplicaciones que podemos desarrollar con software libre.

```
#/usr/bin/env ruby!
require "mysql" # Incluyo en la aplicación la libmysql
connect = Mysql::new("localhost", "root") # Conexión al motor de base de datos mysql
sql = "SHOW DATABASES;" # Sentencia sql que devuelve el listado de Bases de Datos
result = connect.query(sql) # Ejecución del Sql y almacenamiento del resultado
result.each do |regField| # Recorre el listado de registros arrojados
    nombre = regField[0] # Almacena el nombre de las Bases de Datos
    print "* #{nombre} \n" # Muestro por la salida estandar el nombre de la BD
    database = Mysql::new("localhost", "root",nil , nombre); # Me conecto a la BD
    sql = "SHOW TABLES"; # Sql que consulta las tablas dentro de la BD
    tables = database.query(sql) # Ejecución del Sql y almacenamiento del resultado
    tables.each do |regTables| # Recorre el listado de Tablas
        print " - #{regTables[0]}" # Escribo el nombre de la tabla Uno a uno
        sql = "SELECT * FROM #{regTables[0]};" # Sql que consulta todos los registros de la tabla
        count = database.query(sql) # Ejecución del Sql previo y almacenamiento
        print " (#{count.num_rows}) \n" # Excribo el nro de registros de la tabla
```

mysql:

Mysql - Trabajando con Base de Datos

Una de las bases de datos que podemos utilizar con Ruby es Mysql. Esta base de datos esta muy difundida dentro de las aplicaciones que podemos desarrollar con software libre.

```
#/usr/bin/env ruby!
require "mysql" # Incluyo en la aplicación la libmysql
connect = Mysql::new("localhost", "root") # Conexión al motor de base de datos mysql
sql = "SHOW DATABASES;" # Sentencia sql que devuelve el listado de Bases de Datos
result = connect.query(sql) # Ejecución del Sql y almacenamiento del resultado
result.each do |regField| # Recorre el listado de registros arrojados
    nombre = regField[0] # Almacena el nombre de las Bases de Datos
    print "* #{nombre} \n" # Muestro por la salida estandar el nombre de la BD
    database = Mysql::new("localhost", "root",nil , nombre); # Me conecto a la BD
    sql = "SHOW TABLES"; # Sql que consulta las tablas dentro de la BD
    tables = database.query(sql) # Ejecución del Sql y almacenamiento del resultado
    tables.each do |regTables| # Recorre el listado de Tablas
        print " - #{regTables[0]}" # Escribo el nombre de la tabla Uno a uno
        sql = "SELECT * FROM #{regTables[0]};" # Sql que consulta todos los registros de la tabla
        count = database.query(sql) # Ejecución del Sql previo y almacenamiento
        print " (#{count.num_rows}) \n" # Excribo el nro de registros de la tabla
 end # Salida del listado de tablas de la BD
```

mysql:

Mysql - Trabajando con Base de Datos

Una de las bases de datos que podemos utilizar con Ruby es Mysql. Esta base de datos esta muy difundida dentro de las aplicaciones que podemos desarrollar con software libre.

```
#/usr/bin/env ruby!
require "mysql" # Incluyo en la aplicación la libmysql
connect = Mysql::new("localhost", "root") # Conexión al motor de base de datos mysql
sql = "SHOW DATABASES;" # Sentencia sql que devuelve el listado de Bases de Datos
result = connect.query(sql) # Ejecución del Sql y almacenamiento del resultado
result.each do |regField| # Recorre el listado de registros arrojados
    nombre = regField[0] # Almacena el nombre de las Bases de Datos
    print "* #{nombre} \n" # Muestro por la salida estandar el nombre de la BD
    database = Mysql::new("localhost", "root",nil , nombre); # Me conecto a la BD
    sql = "SHOW TABLES"; # Sql que consulta las tablas dentro de la BD
    tables = database.query(sql) # Ejecución del Sql y almacenamiento del resultado
    tables.each do |regTables| # Recorre el listado de Tablas
        print " - #{regTables[0]}" # Escribo el nombre de la tabla Uno a uno
        sql = "SELECT * FROM #{regTables[0]};" # Sql que consulta todos los registros de la tabla
        count = database.query(sql) # Ejecución del Sql previo y almacenamiento
        print " (#{count.num_rows}) \n" # Excribo el nro de registros de la tabla
 end # Salida del listado de tablas de la BD
end # Salida del listado de BD
```

GD:

GD - Trabajando con imágenes.

Las librerías Gd sirven para tratamiento de imágenes y se pueden utilizar dentro de cualquier distribución de Gnu/Linux. Podemos trabajar con imágenes tanto para hacer galerías de fotos Gapchas, etc ... Estas librerías nos sirven para generar imágenes colorearlas montar otras imagenes, incluir textos, marcas de agua, redimensionar, etc ... En el caso de Debian el manual se instala en $/usr/share/doc/libgd-ruby1.8/manual.html$ y podemos conocer todas las funciones para trabajar con las librerías. También tienen un gran soporte para Php, c, c++ y otros lenguajes.

Programando c/GD:

1. #/usr/bin/env ruby

Programando c/GD:

```
1. #/usr/bin/env ruby
2.  require "GD"    # Incluye la librerias GD
```

Programando c/GD:

```
1. #/usr/bin/env ruby
2.  require "GD"    # Incluye la librerias GD
3.  imagen = GD::Image.newTrueColor(450, 220)                # Crea la imagen
```

Programando c/GD:

```
1. #/usr/bin/env ruby
2.  require "GD"    # Incluye la librerias GD
3.  imagen = GD::Image.newTrueColor(450, 220)              # Crea la imagen
4.  transparente = imagen.colorAllocate(100, 100, 100)     # Color transparente
```

Programando c/GD:

```
1. #/usr/bin/env ruby
2.  require "GD"    # Incluye la librerias GD
3.  imagen = GD::Image.newTrueColor(450, 220)              # Crea la imagen
4.  transparente = imagen.colorAllocate(100, 100, 100)     # Color transparente
5.  blanco  = imagen.colorAllocate(255, 255, 255)          # Color Blanco
```

Programando c/GD:

```
1. #/usr/bin/env ruby
2.  require "GD"    # Incluye la librerias GD
3.  imagen = GD::Image.newTrueColor(450, 220)              # Crea la imagen
4.  transparente = imagen.colorAllocate(100, 100, 100)     # Color transparente
5.  blanco  = imagen.colorAllocate(255, 255, 255)          # Color Blanco
6.  negro   = imagen.colorAllocate(0, 0, 0)                # Color Negro
```

Programando c/GD:

```
1. #/usr/bin/env ruby
2.  require "GD"    # Incluye la librerias GD
3.  imagen = GD::Image.newTrueColor(450, 220)              # Crea la imagen
4.  transparente = imagen.colorAllocate(100, 100, 100)     # Color transparente
5.  blanco  = imagen.colorAllocate(255, 255, 255)          # Color Blanco
6.  negro   = imagen.colorAllocate(0, 0, 0)                # Color Negro
7.  rojo    = imagen.colorAllocate(140, 0, 0)              # Color Rojo
```

Programando c/GD:

```
1. #/usr/bin/env ruby
2.  require "GD"    # Incluye la librerias GD
3.  imagen = GD::Image.newTrueColor(450, 220)              # Crea la imagen
4.  transparente = imagen.colorAllocate(100, 100, 100)     # Color transparente
5.  blanco  = imagen.colorAllocate(255, 255, 255)          # Color Blanco
6.  negro   = imagen.colorAllocate(0, 0, 0)                # Color Negro
7.  rojo    = imagen.colorAllocate(140, 0, 0)              # Color Rojo
8.  imagen.transparent(transparente)                       # Hacer transparente el color
```

Programando c/GD:

```
1. #/usr/bin/env ruby
2.  require "GD"    # Incluye la librerias GD
3.  imagen = GD::Image.newTrueColor(450, 220)              # Crea la imagen
4.  transparente = imagen.colorAllocate(100, 100, 100)     # Color transparente
5.  blanco  = imagen.colorAllocate(255, 255, 255)          # Color Blanco
6.  negro   = imagen.colorAllocate(0, 0, 0)                # Color Negro
7.  rojo    = imagen.colorAllocate(140, 0, 0)              # Color Rojo
8.  imagen.transparent(transparente)                       # Hacer transparente el color
9.  imagen.interlace = true                                # Imagen entrelazada
```

Programando c/GD:

```
1.  #/usr/bin/env ruby
2.   require "GD"    # Incluye la librerias GD
3.   imagen = GD::Image.newTrueColor(450, 220)              # Crea la imagen
4.   transparente = imagen.colorAllocate(100, 100, 100)    # Color transparente
5.   blanco  = imagen.colorAllocate(255, 255, 255)         # Color Blanco
6.   negro   = imagen.colorAllocate(0, 0, 0)              # Color Negro
7.   rojo    = imagen.colorAllocate(140, 0, 0)            # Color Rojo
8.   imagen.transparent(transparente)                      # Hacer transparente el color
9.   imagen.interlace = true                               # Imagen entrelazada
10.  imagen.fill(0, 0, rojo)                               # Fondo Rojo
```

Programando c/GD:

```
1.  #/usr/bin/env ruby
2.   require "GD"    # Incluye la librerias GD
3.   imagen = GD::Image.newTrueColor(450, 220)              # Crea la imagen
4.   transparente = imagen.colorAllocate(100, 100, 100)    # Color transparente
5.   blanco  = imagen.colorAllocate(255, 255, 255)         # Color Blanco
6.   negro   = imagen.colorAllocate(0, 0, 0)              # Color Negro
7.   rojo    = imagen.colorAllocate(140, 0, 0)            # Color Rojo
8.   imagen.transparent(transparente)                      # Hacer transparente el color
9.   imagen.interlace = true                               # Imagen entrelazada
10.  imagen.fill(0, 0, rojo)                              # Fondo Rojo
11.  logo = File.open("logo-ejemplo05.jpeg", "rb")        # Abrir archivo c/logo
```

Programando c/GD:

```
1.  #/usr/bin/env ruby
2.   require "GD"    # Incluye la librerias GD
3.   imagen = GD::Image.newTrueColor(450, 220)              # Crea la imagen
4.   transparente = imagen.colorAllocate(100, 100, 100)    # Color transparente
5.   blanco  = imagen.colorAllocate(255, 255, 255)         # Color Blanco
6.   negro   = imagen.colorAllocate(0, 0, 0)              # Color Negro
7.   rojo    = imagen.colorAllocate(140, 0, 0)            # Color Rojo
8.   imagen.transparent(transparente)                      # Hacer transparente el color
9.   imagen.interlace = true                               # Imagen entrelazada
10.  imagen.fill(0, 0, rojo)                               # Fondo Rojo
11.  logo = File.open("logo-ejemplo05.jpeg", "rb")         # Abrir archivo c/logo
12.  image = GD::Image.newFromJpeg(logo)                   # Crearndo imagen con el logo
```

Programando c/GD:

```
1.  #/usr/bin/env ruby
2.   require "GD"    # Incluye la librerias GD
3.   imagen = GD::Image.newTrueColor(450, 220)              # Crea la imagen
4.   transparente = imagen.colorAllocate(100, 100, 100)    # Color transparente
5.   blanco  = imagen.colorAllocate(255, 255, 255)         # Color Blanco
6.   negro   = imagen.colorAllocate(0, 0, 0)              # Color Negro
7.   rojo    = imagen.colorAllocate(140, 0, 0)            # Color Rojo
8.   imagen.transparent(transparente)                      # Hacer transparente el color
9.   imagen.interlace = true                               # Imagen entrelazada
10.  imagen.fill(0, 0, rojo)                               # Fondo Rojo
11.  logo = File.open("logo-ejemplo05.jpeg", "rb")         # Abrir archivo c/logo
12.  image = GD::Image.newFromJpeg(logo)                   # Crearndo imagen con el logo
13.  # Copia y redimensiona el logo en la imagen principal
```

Programando c/GD:

```
1.  #/usr/bin/env ruby
2.   require "GD"    # Incluye la librerias GD
3.   imagen = GD::Image.newTrueColor(450, 220)              # Crea la imagen
4.   transparente = imagen.colorAllocate(100, 100, 100)    # Color transparente
5.   blanco  = imagen.colorAllocate(255, 255, 255)         # Color Blanco
6.   negro   = imagen.colorAllocate(0, 0, 0)              # Color Negro
7.   rojo    = imagen.colorAllocate(140, 0, 0)            # Color Rojo
8.   imagen.transparent(transparente)                      # Hacer transparente el color
9.   imagen.interlace = true                               # Imagen entrelazada
10.  imagen.fill(0, 0, rojo)                               # Fondo Rojo
11.  logo = File.open("logo-ejemplo05.jpeg", "rb")         # Abrir archivo c/logo
12.  image = GD::Image.newFromJpeg(logo)                   # Crearndo imagen con el logo
13.  # Copia y redimensiona el logo en la imagen principal
14.
     image.copyResized(imagen, 220, 50, 0, 0, (image.width/1.5), (image.height/1.5), image.width, image.heig
```

Programando c/GD:

```
1.  #/usr/bin/env ruby
2.   require "GD"    # Incluye la librerias GD
3.   imagen = GD::Image.newTrueColor(450, 220)              # Crea la imagen
4.   transparente = imagen.colorAllocate(100, 100, 100)    # Color transparente
5.   blanco  = imagen.colorAllocate(255, 255, 255)         # Color Blanco
6.   negro   = imagen.colorAllocate(0, 0, 0)              # Color Negro
7.   rojo    = imagen.colorAllocate(140, 0, 0)            # Color Rojo
8.   imagen.transparent(transparente)                      # Hacer transparente el color
9.   imagen.interlace = true                               # Imagen entrelazada
10.  imagen.fill(0, 0, rojo)                               # Fondo Rojo
11.  logo = File.open("logo-ejemplo05.jpeg", "rb")         # Abrir archivo c/logo
12.  image = GD::Image.newFromJpeg(logo)                   # Crearndo imagen con el logo
13.  # Copia y redimensiona el logo en la imagen principal
14.
     image.copyResized(imagen, 220, 50, 0, 0, (image.width/1.5), (image.height/1.5), image.width, image.heig
15.  logo.close # Cierre del archivo c/logo
```

Programando c/GD:

```
1.  #/usr/bin/env ruby
2.   require "GD"    # Incluye la librerias GD
3.   imagen = GD::Image.newTrueColor(450, 220)              # Crea la imagen
4.   transparente = imagen.colorAllocate(100, 100, 100)    # Color transparente
5.   blanco  = imagen.colorAllocate(255, 255, 255)         # Color Blanco
6.   negro   = imagen.colorAllocate(0, 0, 0)              # Color Negro
7.   rojo    = imagen.colorAllocate(140, 0, 0)            # Color Rojo
8.   imagen.transparent(transparente)                      # Hacer transparente el color
9.   imagen.interlace = true                               # Imagen entrelazada
10.  imagen.fill(0, 0, rojo)                               # Fondo Rojo
11.  logo = File.open("logo-ejemplo05.jpeg", "rb")         # Abrir archivo c/logo
12.  image = GD::Image.newFromJpeg(logo)                   # Crearndo imagen con el logo
13.  # Copia y redimensiona el logo en la imagen principal
14.
     image.copyResized(imagen, 220, 50, 0, 0, (image.width/1.5), (image.height/1.5), image.width, image.heig
15.  logo.close # Cierre del archivo c/logo
16.  image.destroy # Destruye el Objeto contenedor del logo
```

Programando c/GD:

```
1.  #/usr/bin/env ruby
2.   require "GD"    # Incluye la librerias GD
3.   imagen = GD::Image.newTrueColor(450, 220)              # Crea la imagen
4.   transparente = imagen.colorAllocate(100, 100, 100)    # Color transparente
5.   blanco  = imagen.colorAllocate(255, 255, 255)         # Color Blanco
6.   negro   = imagen.colorAllocate(0, 0, 0)              # Color Negro
7.   rojo    = imagen.colorAllocate(140, 0, 0)            # Color Rojo
8.   imagen.transparent(transparente)                      # Hacer transparente el color
9.   imagen.interlace = true                               # Imagen entrelazada
10.  imagen.fill(0, 0, rojo)                               # Fondo Rojo
11.  logo = File.open("logo-ejemplo05.jpeg", "rb")         # Abrir archivo c/logo
12.  image = GD::Image.newFromJpeg(logo)                   # Crearndo imagen con el logo
13.  # Copia y redimensiona el logo en la imagen principal
14.
     image.copyResized(imagen, 220, 50, 0, 0, (image.width/1.5), (image.height/1.5), image.width, image.heig
15.  logo.close # Cierre del archivo c/logo
16.  image.destroy # Destruye el Objeto contenedor del logo
```

Programando c/GD:

1. =begin

Programando c/GD:

1. =begin
2. Escribe el texto en la imagen, 1° La sombra Negra

Programando c/GD:

```
1. =begin
2. Escribe el texto en la imagen, 1° La sombra Negra
3. =end
```

Programando c/GD:

```
1. =begin
2. Escribe el texto en la imagen, 1° La sombra Negra
3. =end
4. imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 62,"Programando")
```

Programando c/GD:

```
1. =begin
2. Escribe el texto en la imagen, 1° La sombra Negra
3. =end
4. imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 62,"Programando")
5. imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 122, "con")
```

Programando c/GD:

```
1. =begin
2. Escribe el texto en la imagen, 1° La sombra Negra
3. =end
4. imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 62,"Programando")
5. imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 122, "con")
6. imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 182, "Ruby")
```

Programando c/GD:

```
1. =begin
2. Escribe el texto en la imagen, 1° La sombra Negra
3. =end
4. imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 62,"Programando")
5. imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 122, "con")
6. imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 182, "Ruby")
7. =begin
```

Programando c/GD:

```
1. =begin
2. Escribe el texto en la imagen, 1° La sombra Negra
3. =end
4. imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 62,"Programando")
5. imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 122, "con")
6. imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 182, "Ruby")
7. =begin
8. Escribe el texto en la imagen, 2° El texto transparente
```

Programando c/GD:

```
1. =begin
2. Escribe el texto en la imagen, 1° La sombra Negra
3. =end
4. imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 62,"Programando")
5. imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 122, "con")
6. imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 182, "Ruby")
7. =begin
8. Escribe el texto en la imagen, 2° El texto transparente
9. =end
```

Programando c/GD:

```
1.  =begin
2.  Escribe el texto en la imagen, 1° La sombra Negra
3.  =end
4.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 62,"Programando")
5.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 122, "con")
6.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 182, "Ruby")
7.  =begin
8.  Escribe el texto en la imagen, 2° El texto transparente
9.  =end
10. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 60,"Programando")
```

Programando c/GD:

```
1.  =begin
2.  Escribe el texto en la imagen, 1° La sombra Negra
3.  =end
4.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 62,"Programando")
5.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 122, "con")
6.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 182, "Ruby")
7.  =begin
8.  Escribe el texto en la imagen, 2° El texto transparente
9.  =end
10. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 60,"Programando")
11. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 120, "con")
```

Programando c/GD:

```
1.  =begin
2.  Escribe el texto en la imagen, 1° La sombra Negra
3.  =end
4.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 62,"Programando")
5.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 122, "con")
6.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 182, "Ruby")
7.  =begin
8.  Escribe el texto en la imagen, 2° El texto transparente
9.  =end
10. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 60,"Programando")
11. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 120, "con")
12. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 180, "Ruby")
```

Programando c/GD:

```
1.  =begin
2.  Escribe el texto en la imagen, 1° La sombra Negra
3.  =end
4.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 62,"Programando")
5.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 122, "con")
6.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 182, "Ruby")
7.  =begin
8.  Escribe el texto en la imagen, 2° El texto transparente
9.  =end
10. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 60,"Programando")
11. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 120, "con")
12. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 180, "Ruby")
13. =begin
```

Programando c/GD:

```
1.  =begin
2.  Escribe el texto en la imagen, 1° La sombra Negra
3.  =end
4.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 62,"Programando")
5.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 122, "con")
6.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 182, "Ruby")
7.  =begin
8.  Escribe el texto en la imagen, 2° El texto transparente
9.  =end
10. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 60,"Programando")
11. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 120, "con")
12. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 180, "Ruby")
13. =begin
14. Finalizando la escritura del texto
```

Programando c/GD:

```
1.  =begin
2.  Escribe el texto en la imagen, 1° La sombra Negra
3.  =end
4.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 62,"Programando")
5.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 122, "con")
6.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 182, "Ruby")
7.  =begin
8.  Escribe el texto en la imagen, 2° El texto transparente
9.  =end
10. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 60,"Programando")
11. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 120, "con")
12. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 180, "Ruby")
13. =begin
14. Finalizando la escritura del texto
15. =end
```

Programando c/GD:

```
1.  =begin
2.  Escribe el texto en la imagen, 1° La sombra Negra
3.  =end
4.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 62,"Programando")
5.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 122, "con")
6.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 182, "Ruby")
7.  =begin
8.  Escribe el texto en la imagen, 2° El texto transparente
9.  =end
10. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 60,"Programando")
11. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 120, "con")
12. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 180, "Ruby")
13. =begin
14. Finalizando la escritura del texto
15. =end
16. archivo = open("ejemplo05.png", "wb") # Abriendo el archivo de salida para escritura
```

Programando c/GD:

```
1.  =begin
2.  Escribe el texto en la imagen, 1° La sombra Negra
3.  =end
4.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 62,"Programando")
5.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 122, "con")
6.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 182, "Ruby")
7.  =begin
8.  Escribe el texto en la imagen, 2° El texto transparente
9.  =end
10. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 60,"Programando")
11. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 120, "con")
12. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 180, "Ruby")
13. =begin
14. Finalizando la escritura del texto
15. =end
16. archivo = open("ejemplo05.png", "wb") # Abriendo el archivo de salida para escritura
17. imagen.png archivo # Escribe el archivo de saida
```

Programando c/GD:

```
1.  =begin
2.  Escribe el texto en la imagen, 1° La sombra Negra
3.  =end
4.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 62,"Programando")
5.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 122, "con")
6.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 182, "Ruby")
7.  =begin
8.  Escribe el texto en la imagen, 2° El texto transparente
9.  =end
10. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 60,"Programando")
11. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 120, "con")
12. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 180, "Ruby")
13. =begin
14. Finalizando la escritura del texto
15. =end
16. archivo = open("ejemplo05.png", "wb") # Abriendo el archivo de salida para escritura
17. imagen.png archivo # Escribe el archivo de saida
18. archivo.close # Cierre del archivo de salida
```

Programando c/GD:

```
1.  =begin
2.  Escribe el texto en la imagen, 1° La sombra Negra
3.  =end
4.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 62,"Programando")
5.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 122, "con")
6.  imagen.stringTTF(negro, "./BabelSans.ttf" , 36, 0, 18, 182, "Ruby")
7.  =begin
8.  Escribe el texto en la imagen, 2° El texto transparente
9.  =end
10. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 60,"Programando")
11. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 120, "con")
12. imagen.stringTTF(transparente, "./BabelSans.ttf" , 36, 0, 20, 180, "Ruby")
13. =begin
14. Finalizando la escritura del texto
15. =end
16. archivo = open("ejemplo05.png", "wb") # Abriendo el archivo de salida para escritura
17. imagen.png archivo # Escribe el archivo de saida
18. archivo.close # Cierre del archivo de salida
19. imagen.destroy # Destruye la imagen
```

Bibliografía consultada para realizar la charla:

- Addison Wesley - The Ruby Way 2nd Edition Oct 2006.

Bibliografía consultada para realizar la charla:

- Addison Wesley - The Ruby Way 2nd Edition Oct 2006.
- Programando en Ruby - La Guía de los "Programadores Pragmáticos".

Bibliografía consultada para realizar la charla:

- Addison Wesley - The Ruby Way 2nd Edition Oct 2006.
- Programando en Ruby - La Guía de los "Programadores Pragmáticos".
- Guia del usuario de Ruby.

Bibliografía consultada para realizar la charla:

- Addison Wesley - The Ruby Way 2nd Edition Oct 2006.
- Programando en Ruby - La Guía de los "Programadores Pragmáticos".
- Guia del usuario de Ruby.
- Ruby Developer's Guide.

Bibliografía consultada para realizar la charla:

- Addison Wesley - The Ruby Way 2nd Edition Oct 2006.
- Programando en Ruby - La Guía de los "Programadores Pragmáticos".
- Guia del usuario de Ruby.
- Ruby Developer's Guide.
- Ruby Cookbook O'Relly.

Bibliografía consultada para realizar la charla:

- Addison Wesley - The Ruby Way 2nd Edition Oct 2006.
- Programando en Ruby - La Guía de los "Programadores Pragmáticos".
- Guia del usuario de Ruby.
- Ruby Developer's Guide.
- Ruby Cookbook O'Relly.
- Wikipedia, la enciclopedia libre.

Muchas gracias.

- A todos uds. por presenciar la charla.

Muchas gracias.

- A todos uds. por presenciar la charla.
- A la Facultad de Ciencia y Tecnología de la Uader.

Muchas gracias.

- A todos uds. por presenciar la charla.
- A la Facultad de Ciencia y Tecnología de la Uader.
- A la Facultad de Ciencias Agropecuarias y al centro de Estudiantes.

Muchas gracias.

- A todos uds. por presenciar la charla.
- A la Facultad de Ciencia y Tecnología de la Uader.
- A la Facultad de Ciencias Agropecuarias y al centro de Estudiantes.
- Al Lug Oro Verde.

Muchas gracias.

- A todos uds. por presenciar la charla.
- A la Facultad de Ciencia y Tecnología de la Uader.
- A la Facultad de Ciencias Agropecuarias y al centro de Estudiantes.
- Al Lug Oro Verde.
- Happy Hacking.

Namaste.