

# UNIVERSIDAD NACIONAL AUTONOMA DE NICARAGUA



**UNAN - León**

## **FACULTAD DE CIENCIAS Y TECNOLOGIA**

Carrera: Ingeniería en Telemática

Componente: Software como un servicio

Grupo:1

Docente: Erving Montes

Elaborado por:

- Alli Gissiell Herrera Chow.

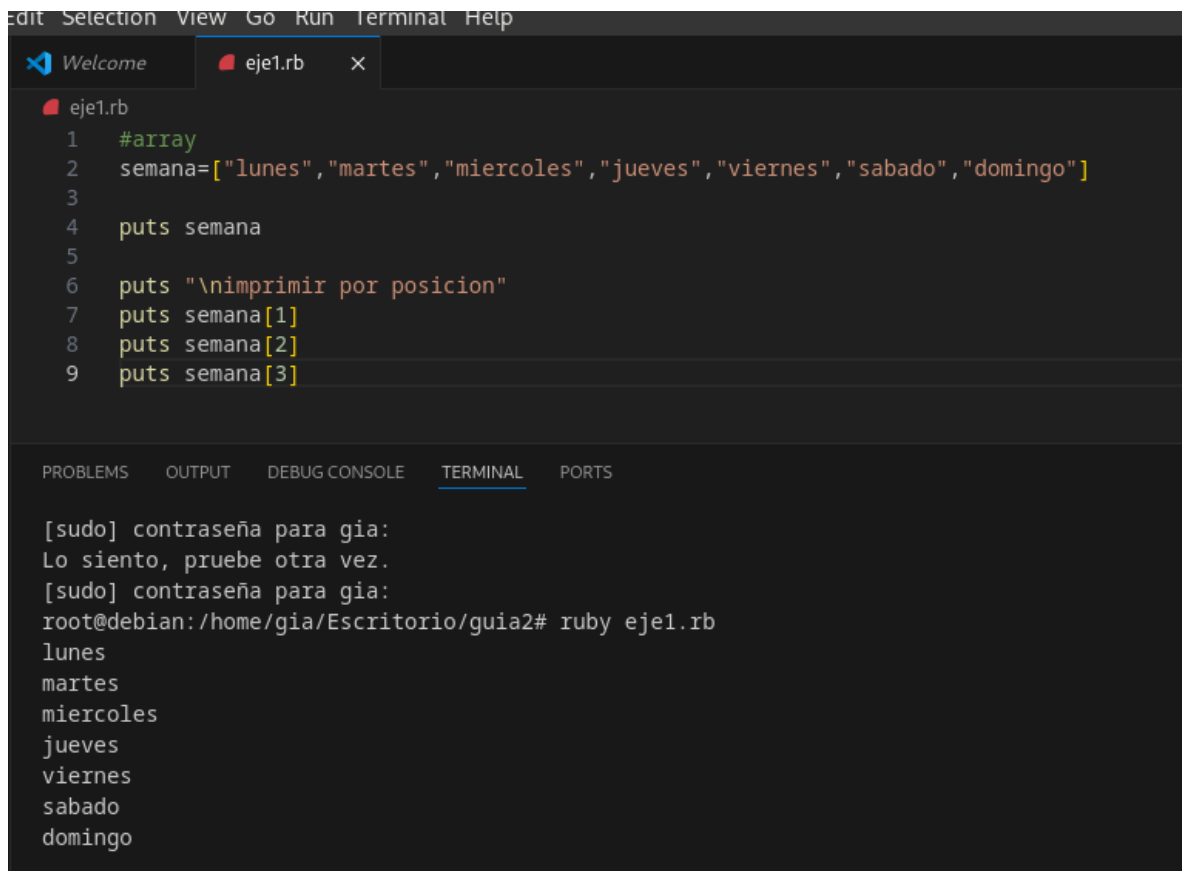
Fecha: 16 de agosto de 2024

**1. Realice cada uno de los enunciados de la guía, probar el funcionamiento y analizar cada uno de los programas planteados.**

1. Array.

1.1. En el directorio ruby, crear un programa en Ruby y asignar a un array los días de la semana, para luego imprimirlos por pantalla.

1.2. Ejecutar el programa en el terminal y analizar lo que imprime.

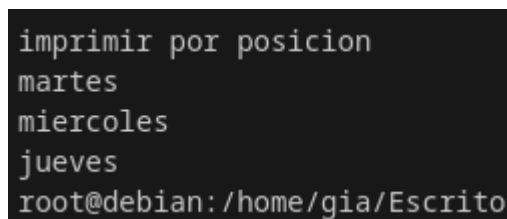


The screenshot shows a code editor with a file named `eje1.rb`. The script defines an array of the days of the week and prints the array, followed by printing elements at specific indices. Below the code, the terminal output shows the execution of the script, displaying the days of the week and the elements at indices 1, 2, and 3.

```
edit Selection View Go Run Terminal Help
Welcome
eje1.rb x
eje1.rb
1 #array
2 semana=["lunes","martes","miercoles","jueves","viernes","sabado","domingo"]
3
4 puts semana
5
6 puts "\nimprimir por posicion"
7 puts semana[1]
8 puts semana[2]
9 puts semana[3]

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

[sudo] contraseña para gia:
Lo siento, pruebe otra vez.
[sudo] contraseña para gia:
root@debian:/home/gia/Escritorio/guia2# ruby eje1.rb
lunes
martes
miercoles
jueves
viernes
sabado
domingo
```



This block is a close-up of the terminal output from the previous screenshot, showing the days of the week and the elements at indices 1, 2, and 3.

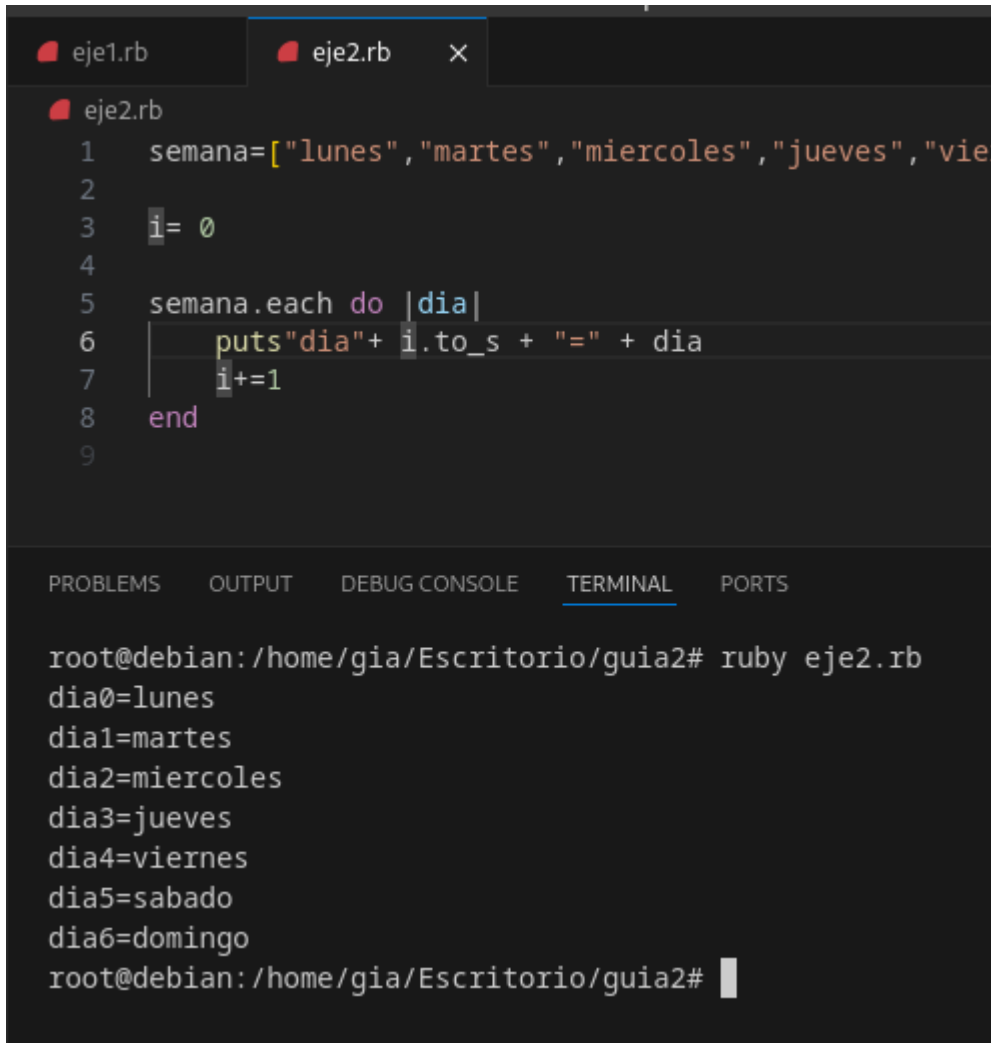
```
imprimir por posicion
martes
miercoles
jueves
root@debian:/home/gia/Escrito
```

2. Método each

2.1. El método each en Ruby se utiliza como iterador para recorrer un array, tomando como ejemplo el programa anterior, crear uno nuevo y utilizar el método each para recorrer el array e imprimirlo por pantalla.

La variable `i` sólo se utiliza como un contador para mostrar que dato es el almacenado en cada posición del array.

## 2.2. Ejecutar el programa y verificar su funcionamiento



The screenshot shows a Ruby IDE with two tabs: `eje1.rb` and `eje2.rb`. The `eje2.rb` tab is active, displaying the following code:

```
1  semana=["lunes","martes","miercoles","jueves","vie  
2  
3  i= 0  
4  
5  semana.each do |dia|  
6  |    puts"dia"+ i.to_s + "=" + dia  
7  |    i+=1  
8  end  
9
```

Below the code editor, the `TERMINAL` tab is selected, showing the execution of the script:

```
root@debian:/home/gia/Escritorio/guia2# ruby eje2.rb  
dia0=lunes  
dia1=martes  
dia2=miercoles  
dia3=jueves  
dia4=viernes  
dia5=sabado  
dia6=domingo  
root@debian:/home/gia/Escritorio/guia2#
```

## 3. Métodos para trabajar con array

En Ruby existen muchos métodos específicamente para trabajar con array, entre los cuales se pueden encontrar: `pop`, `push`, `join`, `last`, `split`. En este enunciado se mostrará el funcionamiento de algunos de ellos, los cuales son muy útiles en el desarrollo de aplicaciones en donde se trabaja con el lenguaje Ruby.

3.1. A continuación, se deberá realizar un programa en el que se utilicen algunos de los métodos antes mencionados

3.2. Ejecutar el programa y verificar el funcionamiento, es importante ver cómo se comporta cada uno de los métodos con respecto al array.

```
eje3.rb
1  semana=["lunes","martes","miercoles","jueves","viernes","sabado","domingo"]
2
3  puts "array en ruby"
4  puts semana
5
6  puts "\nmetodo to_s"
7  puts semana.to_s
8
9  puts "\nmetodo join"
10 puts semana.join(",")
11
```

domingo

metodo to\_s  
["lunes", "martes", "miercoles", "jueves", "viernes", "sabado", "domingo"]

metodo join  
lunes,martes,miercoles,jueves,viernes,sabado,domingo

metodo first  
lunes

metodo last  
domingo

metodo length  
7

No Ports Forwarded ome/gia/Escritorio/guia2#

3.3. Modificar el programa anterior, y hacer uso de los métodos push y pop para ver la diferencia del comportamiento entre ambos, en relación a su uso sobre los arrays.

```
eje3.rb
12 puts semana.length

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

root@debian:/home/gia/Escritorio/guia2# ruby eje3.rb

Array completo
["lunes", "martes", "miercoles", "jueves", "viernes", "sabado", "domingo"]

metodo pop
domingo

metodo length
6

ultimo dato
sabado

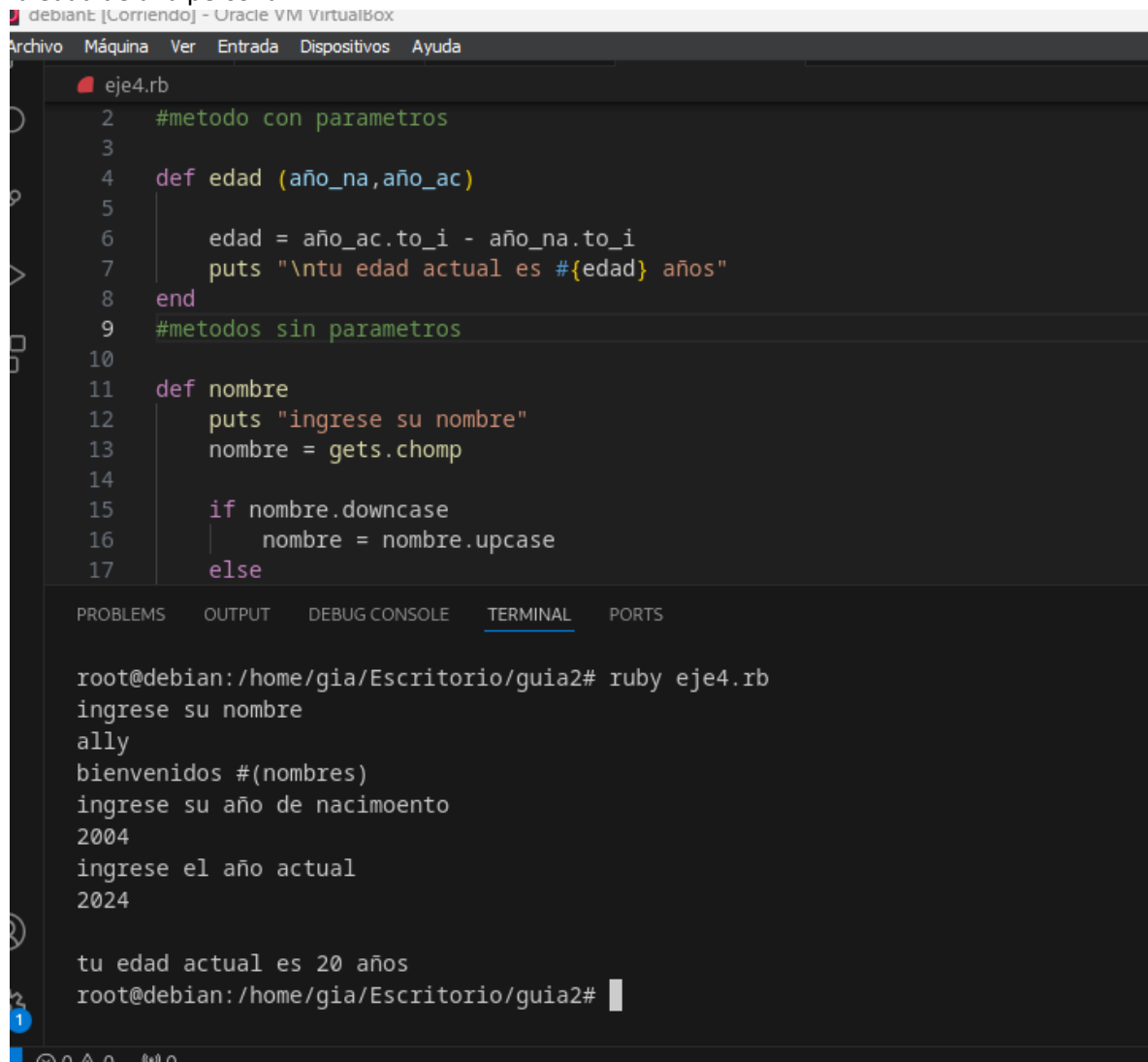
metodo push
lunes
martes
miercoles
jueves
viernes
sabado
final

tamaño nuevo
7
root@debian:/home/gia/Escritorio/guia2#
```

4. Métodos propios 4.1 En Ruby como en cualquier otro lenguaje de programación se pueden definir métodos para que realicen cierto trabajo, para entender un poco mejor de esto, crear un nuevo programa llamado `metodos_propios.rb` y agregar el siguiente código

Como se puede observar se han definido dos métodos, uno llamado `nombre` que no recibe parámetros y el otro llamado `edad`, que recibe dos parámetros y que será el encargado de calcular

la edad de una persona.



The image shows a code editor window titled 'debian [Corriendo] - Oracle VM VirtualBox'. The editor displays a Ruby file named 'eje4.rb' with the following code:

```
1 #ejercicio 4
2 #metodo con parametros
3
4 def edad (año_na,año_ac)
5
6     edad = año_ac.to_i - año_na.to_i
7     puts "\ntu edad actual es #{edad} años"
8 end
9 #metodos sin parametros
10
11 def nombre
12     puts "ingrese su nombre"
13     nombre = gets.chomp
14
15     if nombre.downcase
16         nombre = nombre.upcase
17     else
18         nombre = nombre.downcase
19     end
20 end
```

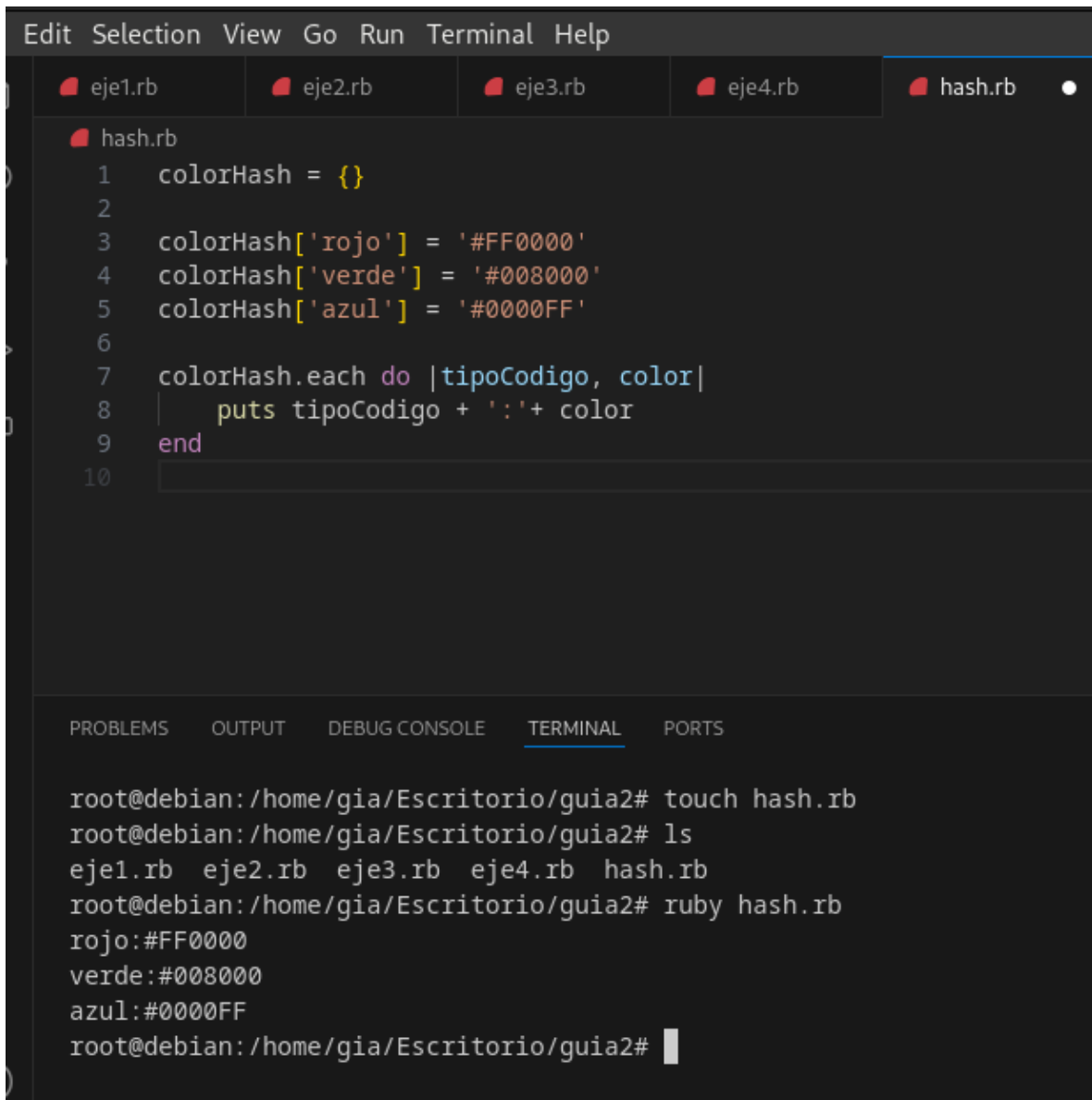
Below the code editor is a terminal window with the following output:

```
root@debian:/home/gia/Escritorio/guia2# ruby eje4.rb
ingrese su nombre
ally
bienvenidos #(nombres)
ingrese su año de nacimiento
2004
ingrese el año actual
2024

tu edad actual es 20 años
root@debian:/home/gia/Escritorio/guia2#
```

5. Hash 5.1. Algo muy utilizado en el lenguaje Ruby son los hashes al momento de trabajar con datos. Crear un programa llamado hash.rb y agregar el código a continuación.

5.2. Ejecutar el programa en el terminal para obtener la salida.



The image shows a code editor window with a dark theme. At the top, there is a menu bar with 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help'. Below the menu bar, there are tabs for 'eje1.rb', 'eje2.rb', 'eje3.rb', 'eje4.rb', and 'hash.rb'. The 'hash.rb' tab is active, showing the following Ruby code:

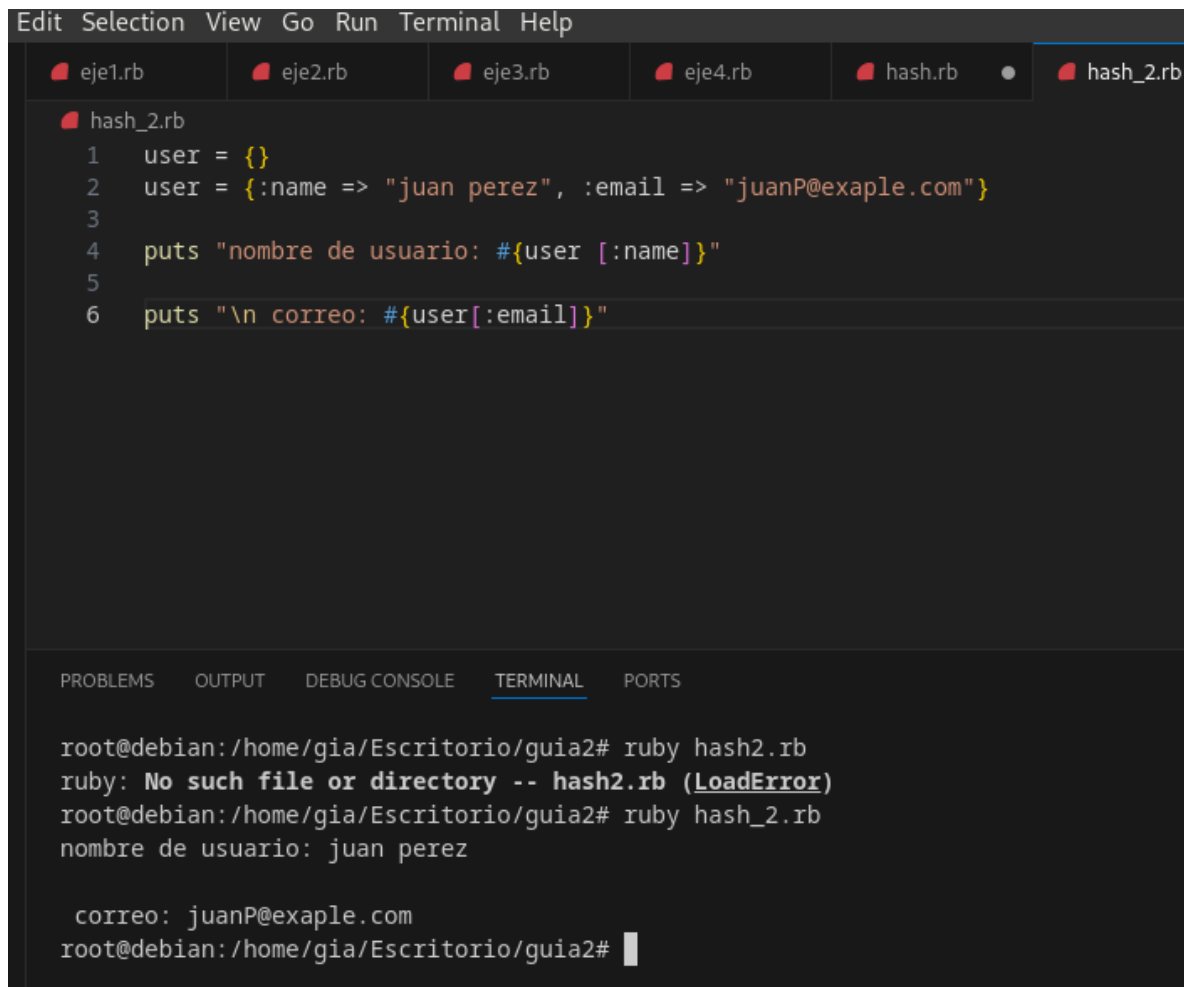
```
1 colorHash = {}
2
3 colorHash['rojo'] = '#FF0000'
4 colorHash['verde'] = '#008000'
5 colorHash['azul'] = '#0000FF'
6
7 colorHash.each do |tipoCodigo, color|
8   puts tipoCodigo + ':' + color
9 end
10
```

Below the code editor, there is a terminal window with tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is active, showing the following commands and output:

```
root@debian:/home/gia/Escritorio/guia2# touch hash.rb
root@debian:/home/gia/Escritorio/guia2# ls
eje1.rb  eje2.rb  eje3.rb  eje4.rb  hash.rb
root@debian:/home/gia/Escritorio/guia2# ruby hash.rb
rojo:#FF0000
verde:#008000
azul:#0000FF
root@debian:/home/gia/Escritorio/guia2#
```

5.3. Para ver de otra manera el funcionamiento, crear un nuevo programa hash\_2.rb y agregar el código

Al ejecutar el programa se observa cómo se hace referencia a los datos del usuario, haciendo uso de la clave para poder mostrarlos por pantalla.



The image shows a screenshot of an IDE with a dark theme. At the top, there is a menu bar with 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help'. Below the menu bar is a tab bar with several tabs: 'eje1.rb', 'eje2.rb', 'eje3.rb', 'eje4.rb', 'hash.rb', and 'hash\_2.rb'. The 'hash\_2.rb' tab is active, showing the following Ruby code:

```
1 user = {}
2 user = {:name => "juan perez", :email => "juanP@exaple.com"}
3
4 puts "nombre de usuario: #{user[:name]}"
5
6 puts "\n correo: #{user[:email]}"
```

Below the code editor is a panel with tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is active, showing the following output:

```
root@debian:/home/gia/Escritorio/guia2# ruby hash2.rb
ruby: No such file or directory -- hash2.rb (LoadError)
root@debian:/home/gia/Escritorio/guia2# ruby hash_2.rb
nombre de usuario: juan perez

correo: juanP@exaple.com
root@debian:/home/gia/Escritorio/guia2#
```

6. Clases 6.1. Crear un programa `clases.rb`, en el cual se creará una clase Palíndromo que contendrá un método para verificar una frase ingresada, como aparece a continuación
- 6.2. Ejecutar el programa e ingresar la palabra "level" para verificar su correcto funcionamiento.



```
class Palindromo

  def verificar_frase(frase)

    if frase == frase.reverse
      puts "la frase #{frase} es palindromo"
    else
      puts "la frase #{frase} no es palindromo"
    end
  end

end

puts "ingrese una frase"
frase = gets.chomp

verificar = Palindromo.new
```

BLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
t@debian:/home/gia/Escritorio/guia2# ruby clases.rb
rese una frase
el
frase level es palindromo
t@debian:/home/gia/Escritorio/guia2#
```

## 7. Variable de instancia.

7.1. Las variables de instancia son variables de un objeto, una de las diferencias de las variables locales es que estas existen hasta que el método ha terminado e inician con arroba "@". Crear un programa en Ruby llamado variables.rb y escribir lo siguiente:

Como se observa en el código la variable `numero_mostrar`, se utiliza en los métodos `rodar` y `mostrar`, y siempre mantiene el mismo el valor. 7.2. Ejecutar el programa y verificar el funcionamiento de la variable `numero_mostrar`, la cual mantiene su valor en todos los métodos hasta ser mostrada por pantalla

```
o Máquina Ver Entrada Dispositivos Ayuda
var.rb
1  class Dado
7    def rodar
8      @numero_mostrar = 1 + rand(6)
9    end
10
11   def mostrar
12     @numero_mostrar
13   end
14 end
15

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

cuantas veces desea lanzar el dado
5

Lanzamiento
4

Lanzamiento
4

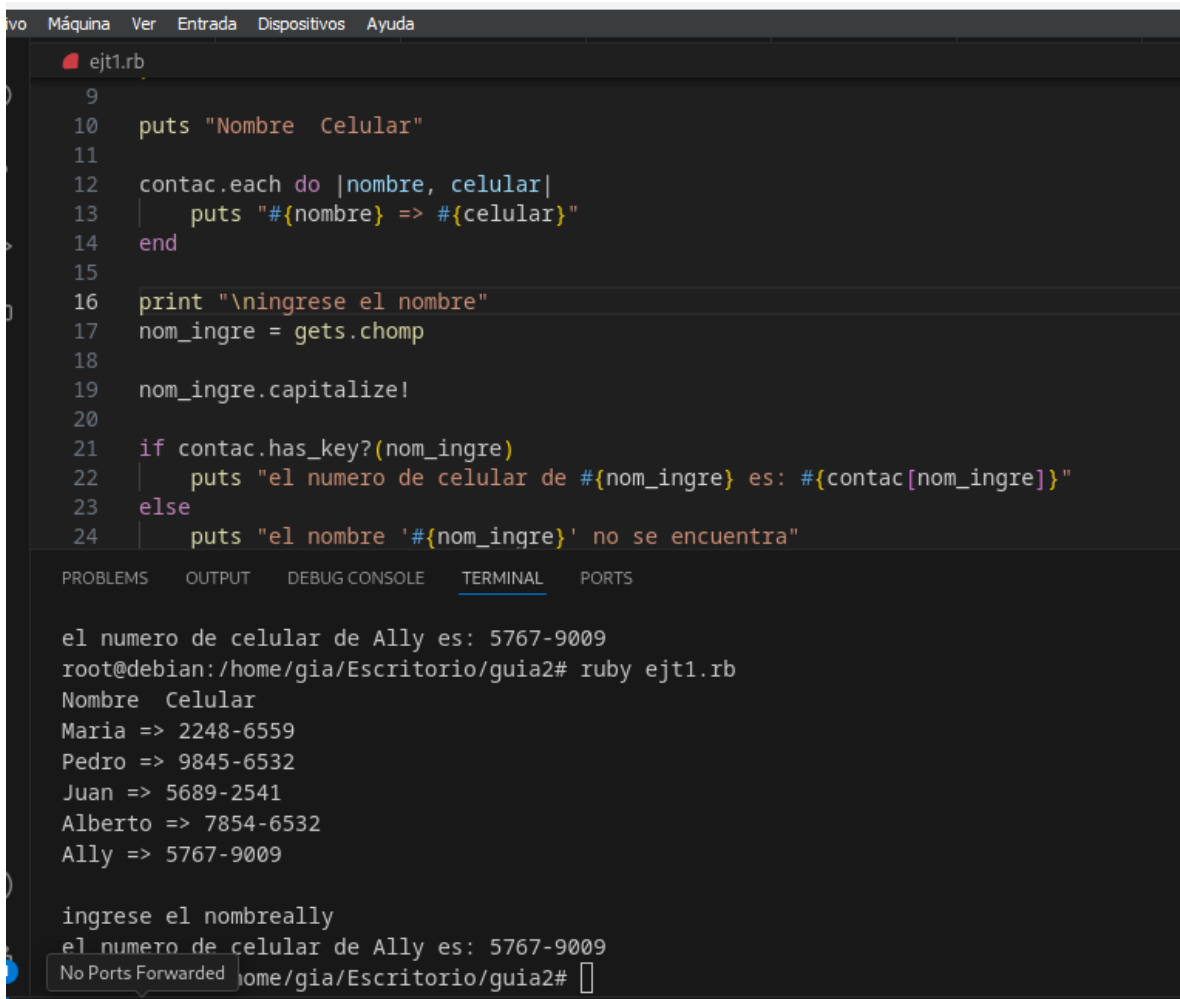
Lanzamiento
3

Lanzamiento
1

Lanzamiento
5

No Ports Forwarded home/gia/Escritorio/guia2#
```

2. Crear un programa en Ruby que contenga un hash, el cual este compuesto de nombre =clave y celular = valor, el programa deberá mostrar el hash completo, solicitar el nombre que sería la clave y retornar el celular que sería el valor, correspondiente a ese nombre. Deberá validar si el dato existe en el hash y que cuando se ingrese un nombre en minúscula a como se muestra en la figura 32, el nombre Juan se ingresó en minúscula y el programa devuelve el celular correspondiente al nombre.



```
ejt1.rb
9
10 puts "Nombre Celular"
11
12 contac.each do |nombre, celular|
13   puts "#{nombre} => #{celular}"
14 end
15
16 print "\ningrese el nombre"
17 nom_ingre = gets.chomp
18
19 nom_ingre.capitalize!
20
21 if contac.has_key?(nom_ingre)
22   puts "el numero de celular de #{nom_ingre} es: #{contac[nom_ingre]}"
23 else
24   puts "el nombre '#{nom_ingre}' no se encuentra"
25 end
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
el numero de celular de Ally es: 5767-9009
root@debian:/home/gia/Escritorio/guia2# ruby ejt1.rb
Nombre Celular
Maria => 2248-6559
Pedro => 9845-6532
Juan => 5689-2541
Alberto => 7854-6532
Ally => 5767-9009

ingrese el nombreally
el numero de celular de Ally es: 5767-9009
No Ports Forwarded home/gia/Escritorio/guia2#
```

3. Realice un programa en Ruby que solicite por pantalla un número cualquiera y que imprima la suma de los números pares e impares que componen el número ingresado, para la solución crear una clase de nombre Calcular la cual contendrá 2 métodos, el primer método para los cálculos de los números pares y el segundo método para los cálculos de los numero impares, se deberá mostrar a como se muestra.

```
ejt2.rb
1  class Calcular
2
3      def initialize(numero)
4          @numero = numero.to_s
5      end
6
7
8      def sum_par
9
10         pares = @numero.chars.select{|char|char.to_i.even? }.map(&:to_i)
11         pares.sum
12     end
13
14
15
16     def suma_impares
17         impares = @numero.chars.select{|char|char.to_i.odd? }.map(&:to_i)
18         impares.sum
19
20
21     end
22
23 end
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
root@debian:/home/gia/Escritorio/guia2# ruby ejt2.rb
ingrese un numero:10
suma de numeros pares:0
suma de numeros impares:1
root@debian:/home/gia/Escritorio/guia2#
```