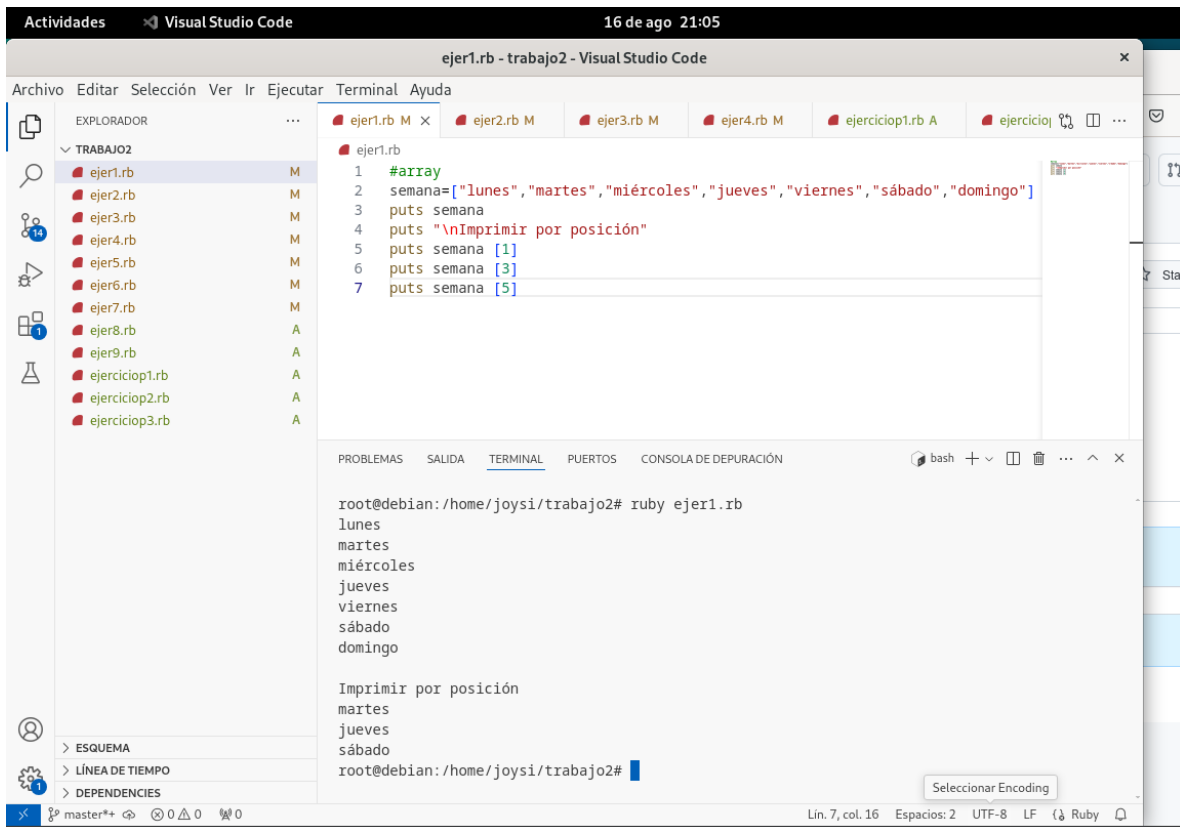


En el directorio ruby, crear un programa en Ruby y asignar a un array los días de la semana, para luego imprimirlos por pantalla. 1.2. Ejecutar el programa en el terminal y analizar lo que imprime.



The screenshot shows the Visual Studio Code interface. On the left, the Explorer panel shows a project named 'TRABAJO2' with several Ruby files. The main editor displays the file 'ejer1.rb' with the following Ruby code:

```
1 #array
2 semana=["lunes","martes","miércoles","jueves","viernes","sábado","domingo"]
3 puts semana
4 puts "\nImprimir por posición"
5 puts semana [1]
6 puts semana [3]
7 puts semana [5]
```

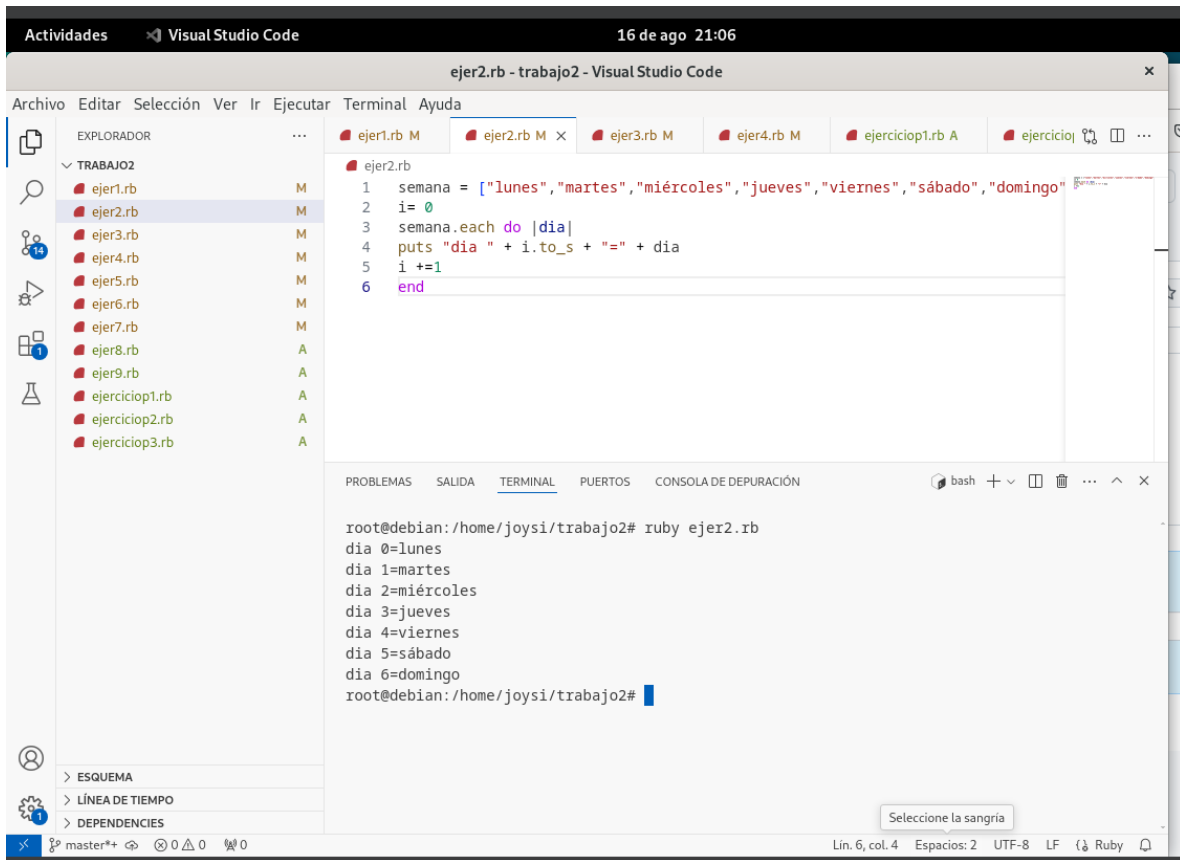
Below the editor, the Terminal panel shows the execution of the program. The prompt is 'root@debian: /home/joysi/trabajo2#'. The command 'ruby ejer1.rb' has been executed, resulting in the following output:

```
lunes
martes
miércoles
jueves
viernes
sábado
domingo

Imprimir por posición
martes
jueves
sábado
root@debian: /home/joysi/trabajo2#
```

The status bar at the bottom indicates the current line and column (Lin. 7, col. 16), the number of spaces (Espacios: 2), the encoding (UTF-8), the line feed (LF), and the language (Ruby).

2. 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.



The screenshot shows the Visual Studio Code interface with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The file explorer shows a project named 'TRABAJO2' with several Ruby files. The code editor displays a file named 'ejer2.rb' with the following Ruby code:

```
1 semana = ["lunes", "martes", "miércoles", "jueves", "viernes", "sábado", "domingo"]
2 i= 0
3 semana.each do |dia|
4   puts "dia " + i.to_s + "=" + dia
5   i +=1
6 end
```

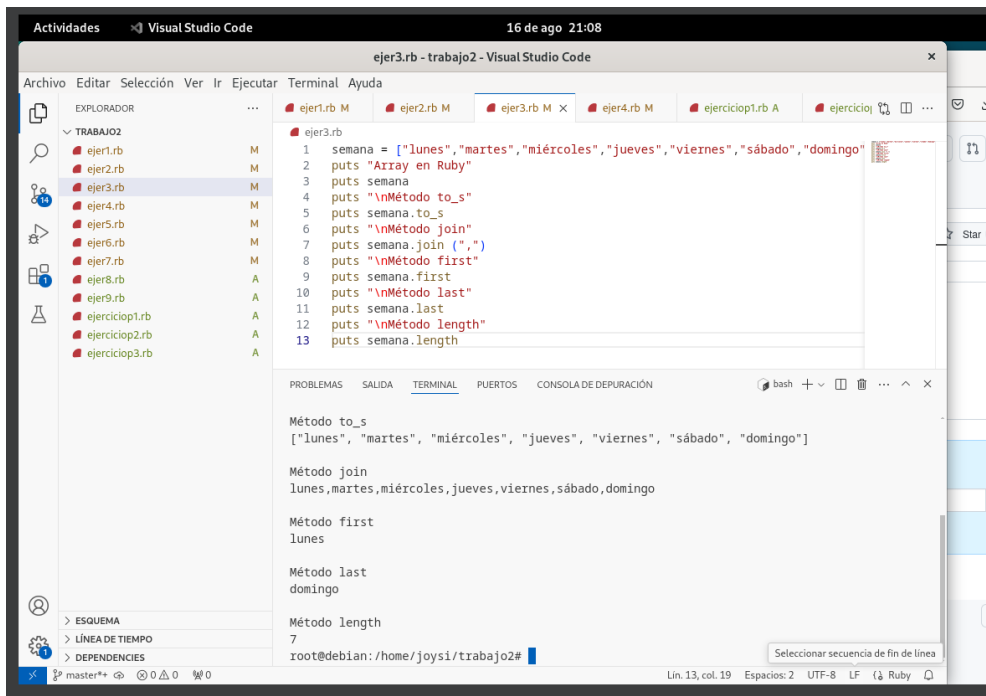
The terminal at the bottom shows the command 'ruby ejer2.rb' being executed, resulting in the following output:

```
root@debian:/home/joysi/trabajo2# ruby ejer2.rb
dia 0=lunes
dia 1=martes
dia 2=miércoles
dia 3=jueves
dia 4=viernes
dia 5=sábado
dia 6=domingo
root@debian:/home/joysi/trabajo2#
```

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.



The screenshot shows the Visual Studio Code interface with a file explorer on the left containing a directory named 'TRABAJO2' with several Ruby files. The main editor displays 'ejer3.rb' with the following code:

```
1 semana = ["lunes","martes","miércoles","jueves","viernes","sábado","domingo"]
2 puts "Array en Ruby"
3 puts semana
4 puts "\nMétodo to_s"
5 puts semana.to_s
6 puts "\nMétodo join"
7 puts semana.join(",")
8 puts "\nMétodo first"
9 puts semana.first
10 puts "\nMétodo last"
11 puts semana.last
12 puts "\nMétodo length"
13 puts semana.length
```

The terminal at the bottom shows the output of the script:

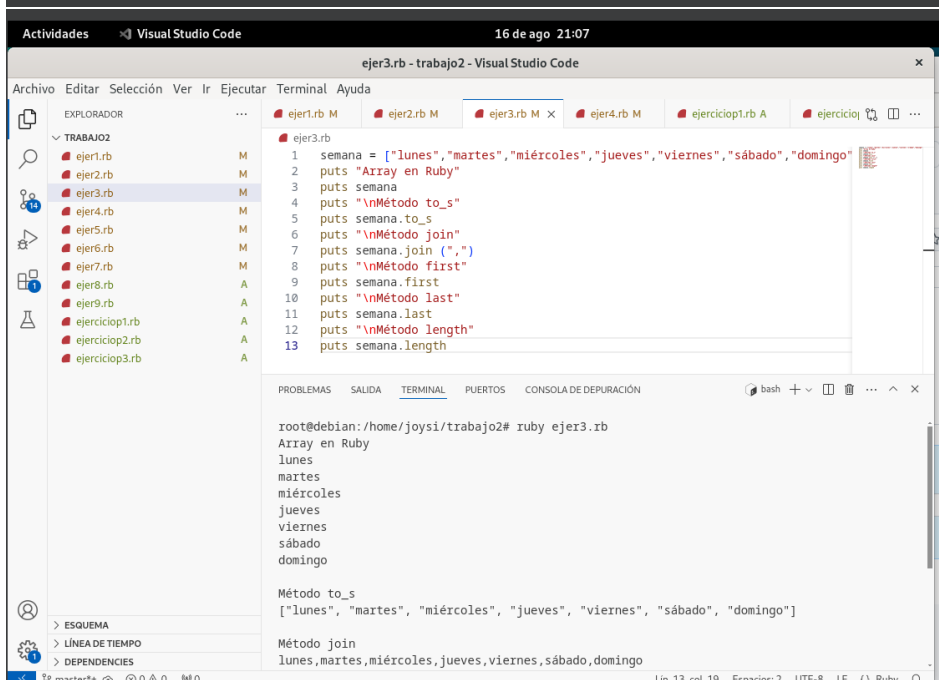
```
Método to_s
["lunes", "martes", "miércoles", "jueves", "viernes", "sábado", "domingo"]

Método join
lunes,martes,miércoles,jueves,viernes,sábado,domingo

Método first
lunes

Método last
domingo

Método length
7
```



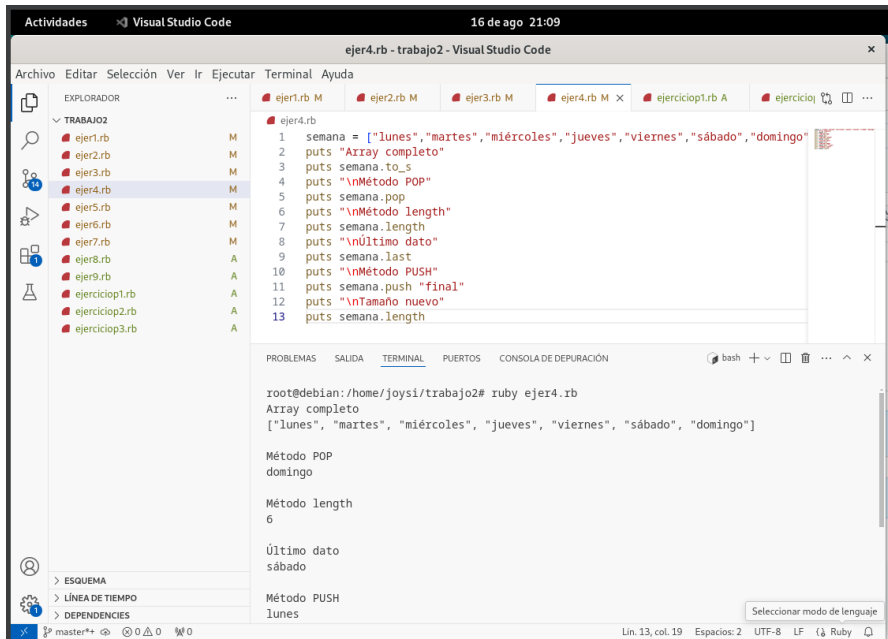
This screenshot shows the same Visual Studio Code interface, but the terminal now shows the output of running the script with the 'ruby' command:

```
root@debian:/home/joysi/trabajo2# ruby ejer3.rb
Array en Ruby
lunes
martes
miércoles
jueves
viernes
sábado
domingo

Método to_s
["lunes", "martes", "miércoles", "jueves", "viernes", "sábado", "domingo"]

Método join
lunes,martes,miércoles,jueves,viernes,sábado,domingo
```

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



```
1 semana = ["lunes", "martes", "miércoles", "jueves", "viernes", "sábado", "domingo"]
2 puts "Array completo"
3 puts semana.to_s
4 puts "\nMétodo POP"
5 puts semana.pop
6 puts "\nMétodo length"
7 puts semana.length
8 puts "\nÚltimo dato"
9 puts semana.last
10 puts "\nMétodo PUSH"
11 puts semana.push "final"
12 puts "\nTamaño nuevo"
13 puts semana.length
```

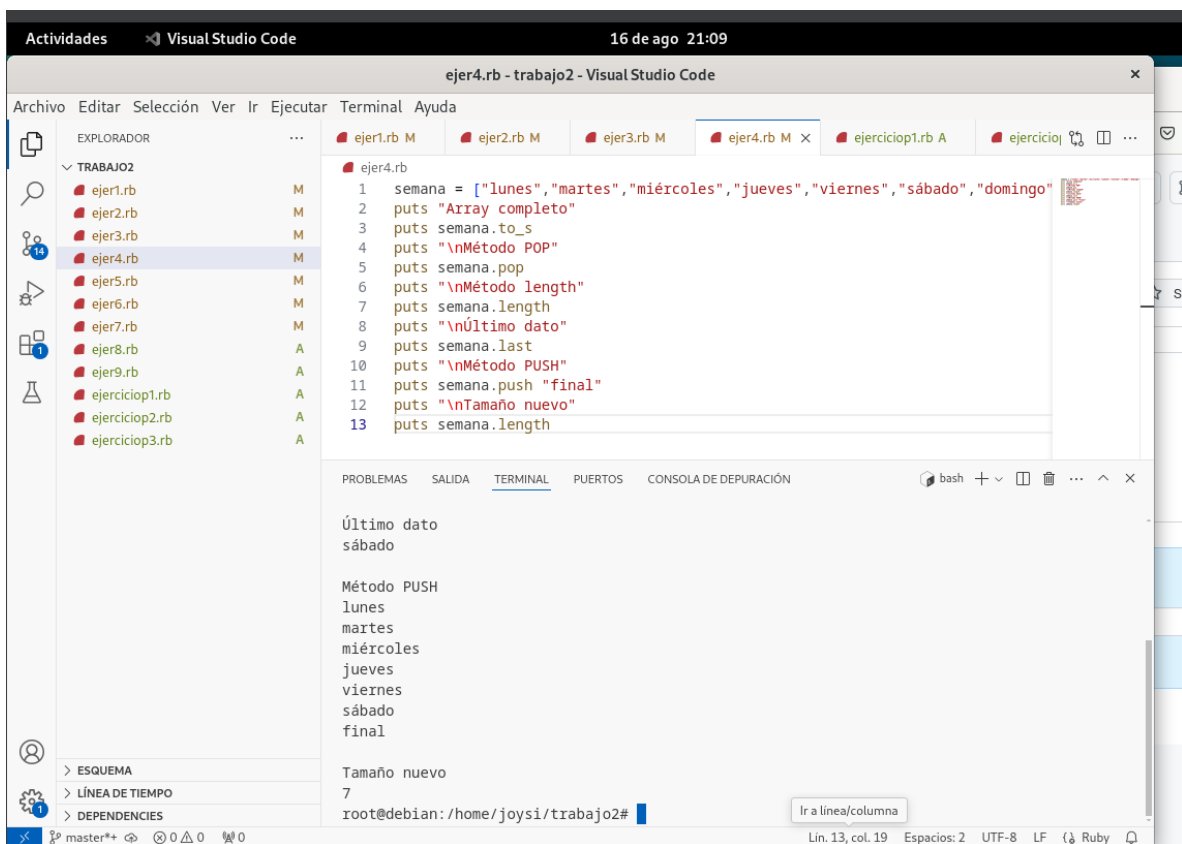
```
root@debian:/home/joysi/trabajo2# ruby ejer4.rb
Array completo
["lunes", "martes", "miércoles", "jueves", "viernes", "sábado", "domingo"]

Método POP
domingo

Método length
6

Último dato
sábado

Método PUSH
lunes
```



```
1 semana = ["lunes", "martes", "miércoles", "jueves", "viernes", "sábado", "domingo"]
2 puts "Array completo"
3 puts semana.to_s
4 puts "\nMétodo POP"
5 puts semana.pop
6 puts "\nMétodo length"
7 puts semana.length
8 puts "\nÚltimo dato"
9 puts semana.last
10 puts "\nMétodo PUSH"
11 puts semana.push "final"
12 puts "\nTamaño nuevo"
13 puts semana.length
```

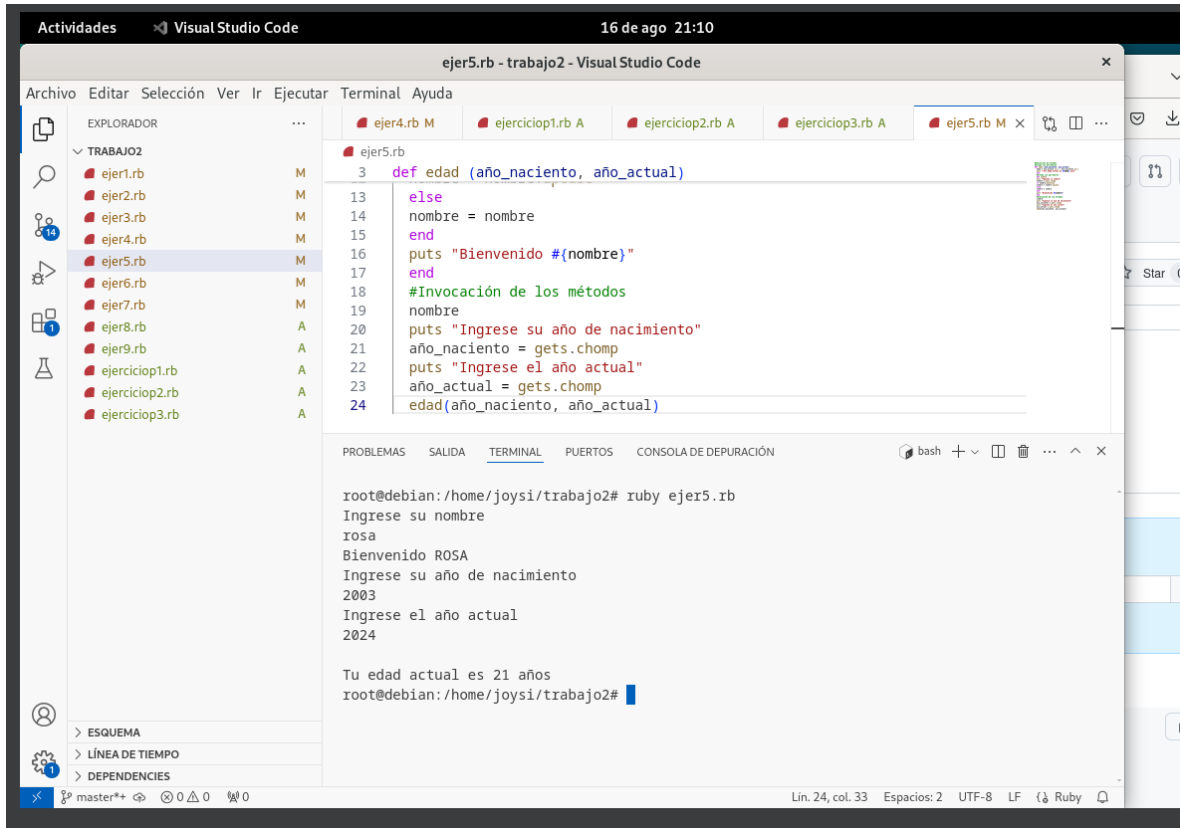
```
Último dato
sábado

Método PUSH
lunes
martes
miércoles
jueves
viernes
sábado
final

Tamaño nuevo
7
root@debian:/home/joysi/trabajo2#
```

Métodos propios

5. 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.



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows a project named 'TRABAJO2' with several files. The file 'ejer5.rb' is selected and open in the editor. The code in 'ejer5.rb' defines a method 'edad' that takes 'año_nacimiento' and 'año_actual' as arguments. It uses a 'rescue' block to handle a 'ZeroDivisionError' and prints a welcome message. The terminal window at the bottom shows the execution of the script, where the user enters 'rosa' for the name, '2003' for the birth year, and '2024' for the current year. The output shows the calculated age of 21 years.

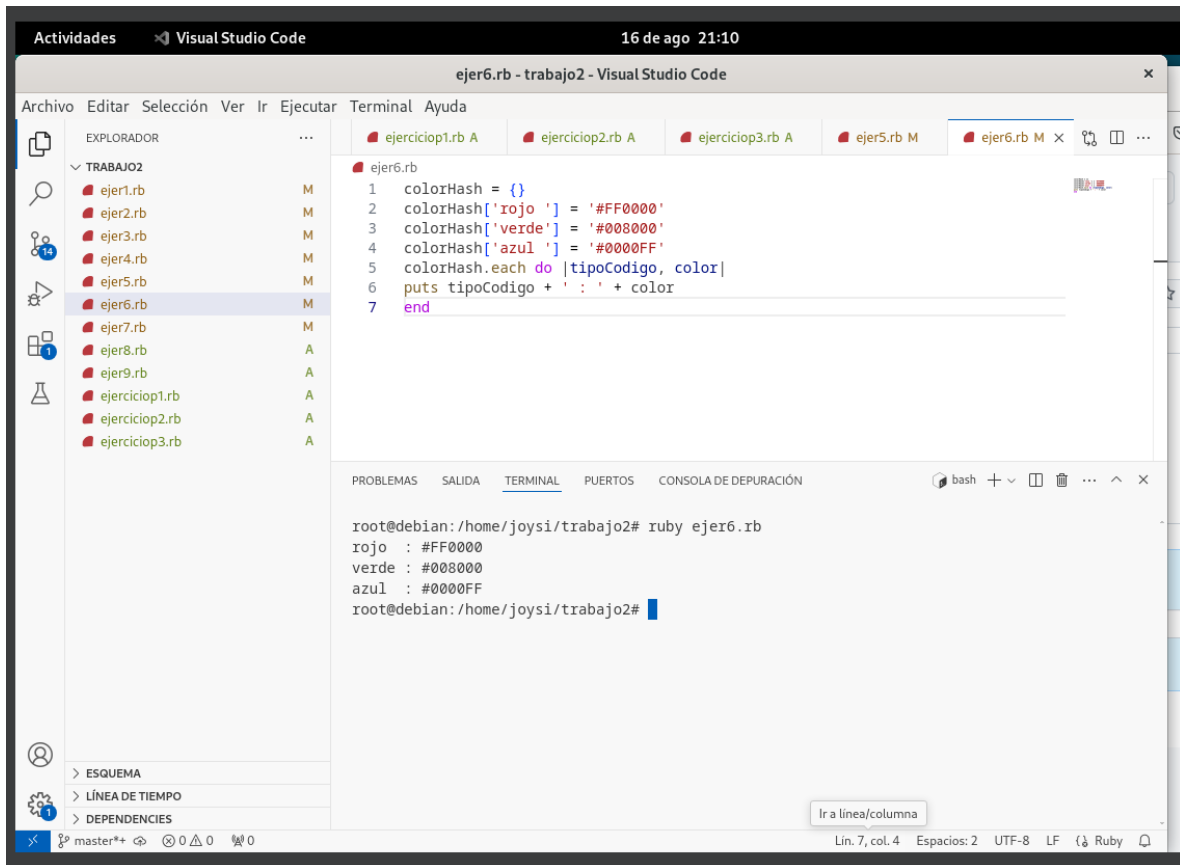
```
ejer5.rb
3 def edad (año_nacimiento, año_actual)
13 rescue ZeroDivisionError => e
14   nombre = nombre
15 end
16 puts "Bienvenido #{nombre}"
17 end
18 #Invocación de los métodos
19 nombre
20 puts "Ingrese su año de nacimiento"
21 año_nacimiento = gets.chomp
22 puts "Ingrese el año actual"
23 año_actual = gets.chomp
24 edad(año_nacimiento, año_actual)
```

```
root@debian:/home/joyisi/trabajo2# ruby ejer5.rb
Ingrese su nombre
rosa
Bienvenido ROSA
Ingrese su año de nacimiento
2003
Ingrese el año actual
2024

Tu edad actual es 21 años
root@debian:/home/joyisi/trabajo2#
```

6. Hash

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.

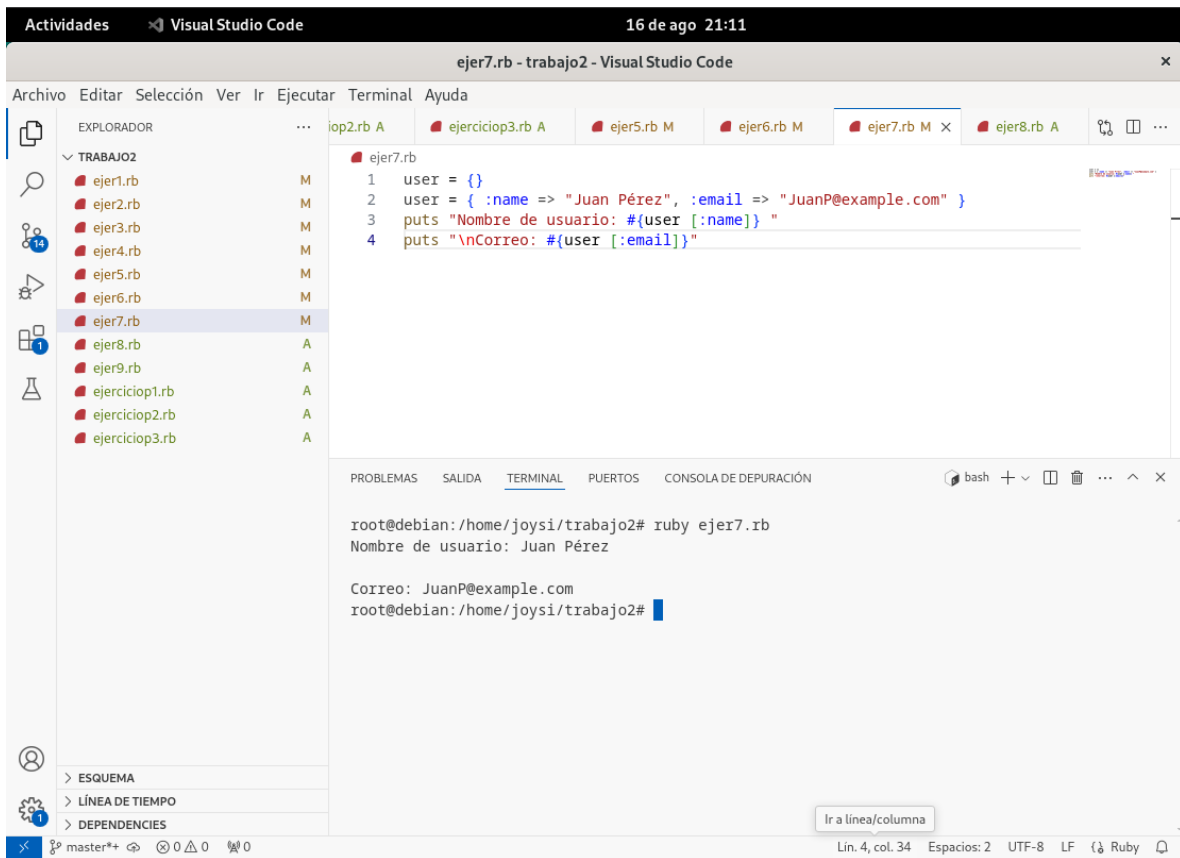


The screenshot shows the Visual Studio Code interface with the file explorer on the left displaying a directory named 'TRABAJO2' containing several Ruby files. The main editor window shows the content of 'ejer6.rb', which defines a hash and iterates over its values. The terminal at the bottom shows the command to run the file and its output.

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

```
root@debian:/home/joyasi/trabajo2# ruby ejer6.rb
rojo : #FF0000
verde : #008000
azul : #0000FF
root@debian:/home/joyasi/trabajo2#
```

7. Para ver de otra manera el funcionamiento, crear un nuevo programa hash_2.rb y agregar el código



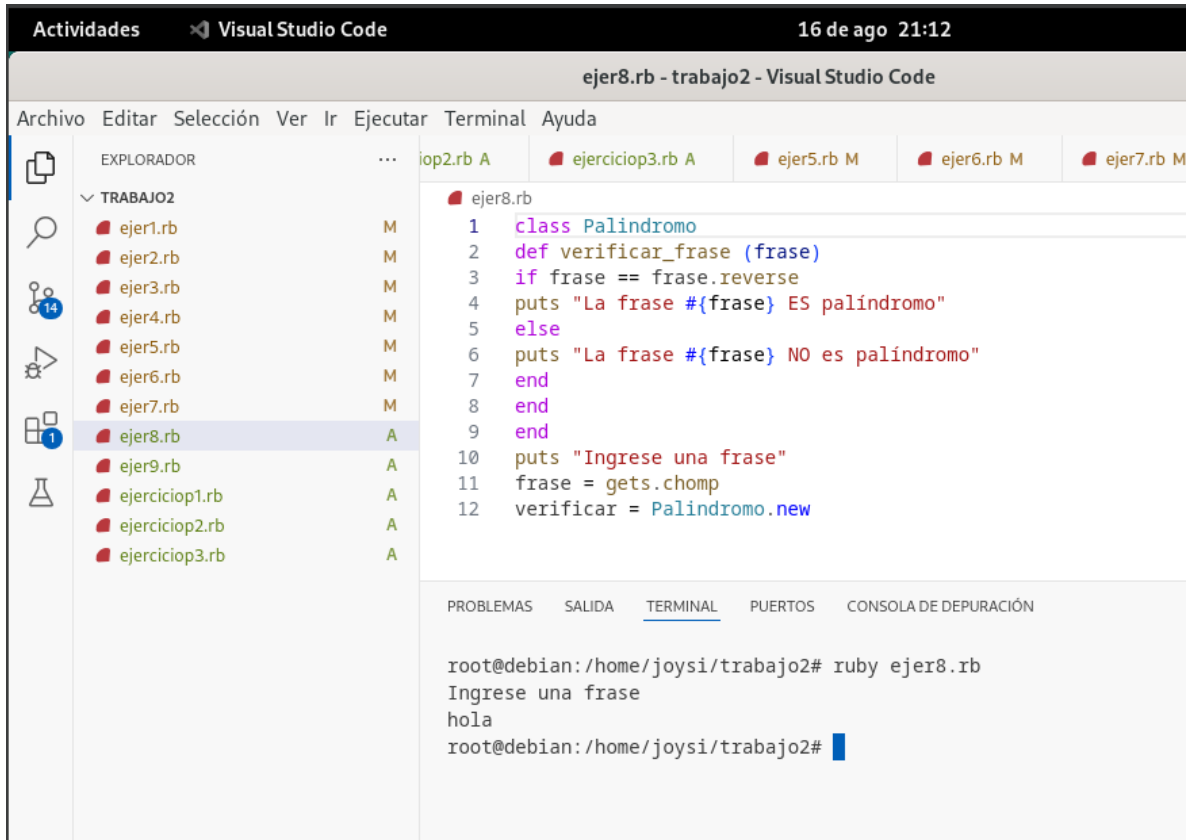
The screenshot shows the Visual Studio Code interface with the file explorer on the left displaying a directory named 'TRABAJO2' containing several Ruby files. The file 'ejer7.rb' is selected. The main editor shows the code for 'ejer7.rb', which defines a hash for a user and prints its details. Below the editor, the terminal window shows the command 'ruby ejer7.rb' being executed, resulting in the user's name and email being printed.

```
Actividades Visual Studio Code 16 de ago 21:11
ej77.rb - trabajo2 - Visual Studio Code
Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda
EXPLORADOR
TRABAJO2
  ejer1.rb M
  ejer2.rb M
  ejer3.rb M
  ejer4.rb M
  ejer5.rb M
  ejer6.rb M
  ejer7.rb M
  ejer8.rb A
  ejer9.rb A
  ejerciciop1.rb A
  ejerciciop2.rb A
  ejerciciop3.rb A
ej77.rb
1 user = {}
2 user = { :name => "Juan Pérez", :email => "JuanP@example.com" }
3 puts "Nombre de usuario: #{user[:name]} "
4 puts "\nCorreo: #{user[:email]}"
PROBLEMAS SALIDA TERMINAL PUERTOS CONSOLA DE DEPURACIÓN
bash
root@debian:/home/joysi/trabajo2# ruby ejer7.rb
Nombre de usuario: Juan Pérez

Correo: JuanP@example.com
root@debian:/home/joysi/trabajo2#
Ir a línea/columna
Lin. 4, col. 34 Espacios: 2 UTF-8 LF Ruby
```

8 Clases

Crear un programa `clases.rb`, en el cual se creará una clase `Palindromo` que contendrá un método para verificar una frase ingresada, como aparece a continuación.



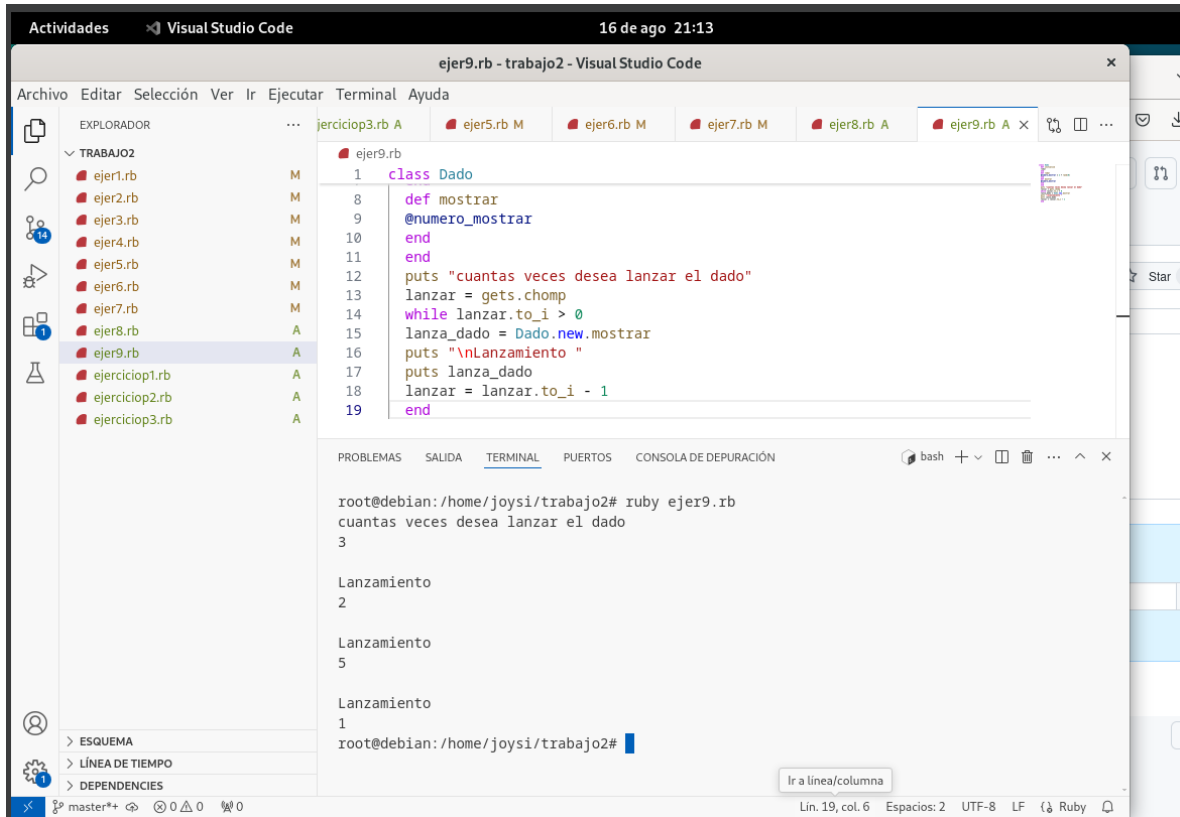
The screenshot shows the Visual Studio Code interface with the file explorer on the left, the editor in the center, and the terminal at the bottom. The file explorer shows a project named 'TRABAJO2' with several files, including 'ejer8.rb' which is selected. The editor displays the code for 'ejer8.rb', which defines a 'Palindromo' class with a 'verificar_frase' method. The terminal shows the command 'ruby ejer8.rb' being executed, followed by the prompt 'Ingrese una frase' and the input 'hola'.

```
Actividades Visual Studio Code 16 de ago 21:12
ejer8.rb - trabajo2 - Visual Studio Code
Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda
EXPLORADOR
TRABAJO2
  ejer1.rb M
  ejer2.rb M
  ejer3.rb M
  ejer4.rb M
  ejer5.rb M
  ejer6.rb M
  ejer7.rb M
  ejer8.rb A
  ejer9.rb A
  ejerciciop1.rb A
  ejerciciop2.rb A
  ejerciciop3.rb A
ejerciciop2.rb A
ejerciciop3.rb A
ejer8.rb
1 class Palindromo
2   def verificar_frase (frase)
3     if frase == frase.reverse
4       puts "La frase #{frase} ES palindromo"
5     else
6       puts "La frase #{frase} NO es palindromo"
7     end
8   end
9 end
10 puts "Ingrese una frase"
11 frase = gets.chomp
12 verificar = Palindromo.new

PROBLEMAS SALIDA TERMINAL PUERTOS CONSOLA DE DEPURACIÓN
root@debian:/home/joysi/trabajo2# ruby ejer8.rb
Ingrese una frase
hola
root@debian:/home/joysi/trabajo2#
```


9. Variable de instancia.

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:



The screenshot shows the Visual Studio Code interface with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The file explorer shows a project named 'TRABAJO2' with several Ruby files. The code editor displays the content of 'ejer9.rb', which defines a 'Dado' class with a 'mostrar' method and a loop that simulates rolling a die. The terminal shows the command 'ruby ejer9.rb' being executed, and the output matches the logic in the code: it asks for the number of rolls, then shows three consecutive rolls with values 3, 2, and 5.

```
1 class Dado
8   def mostrar
9     @numero_mostrar
10    end
11  end
12  puts "cuantas veces desea lanzar el dado"
13  lanzar = gets.chomp
14  while lanzar.to_i > 0
15    lanza_dado = Dado.new.mostrar
16    puts "\nLanzamiento "
17    puts lanza_dado
18    lanzar = lanzar.to_i - 1
19  end
```

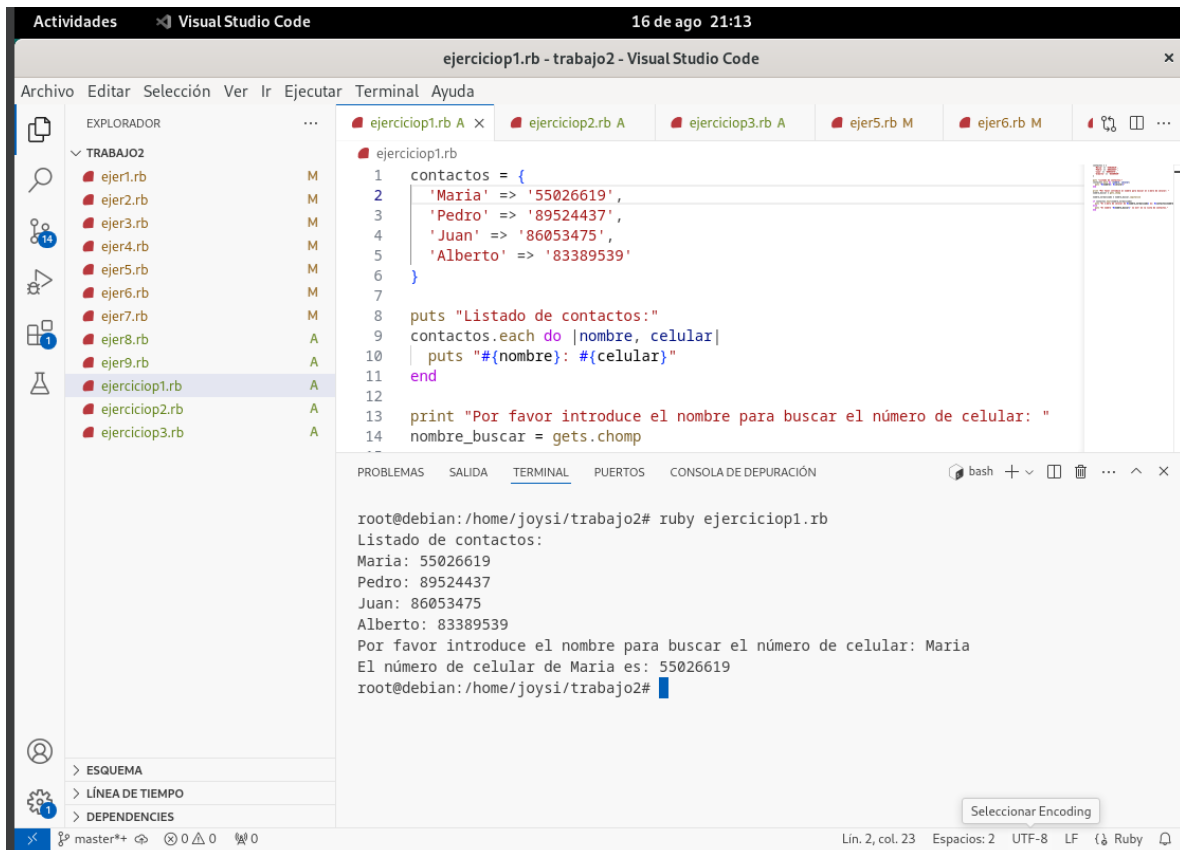
```
root@debian:/home/joysi/trabajo2# ruby ejer9.rb
cuantas veces desea lanzar el dado
3

Lanzamiento
2

Lanzamiento
5

Lanzamiento
1
root@debian:/home/joysi/trabajo2#
```

10. 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.

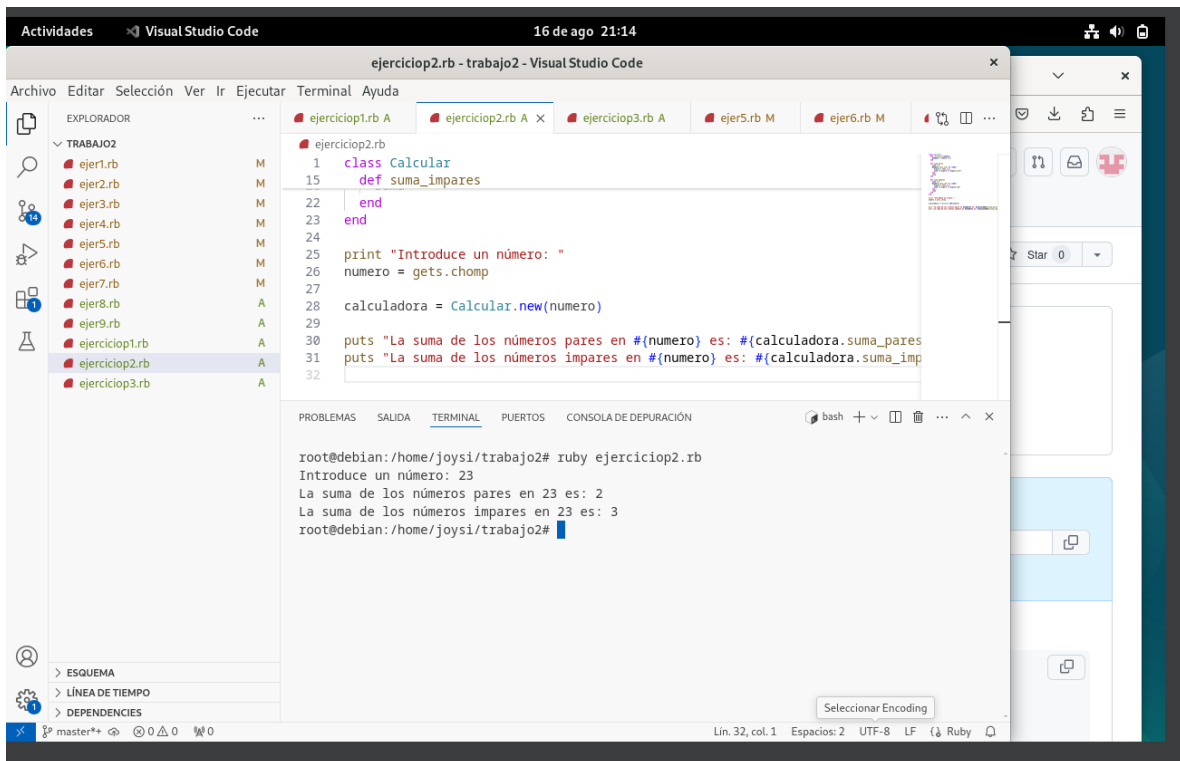


The screenshot shows the Visual Studio Code interface with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The file explorer shows a project named 'TRABAJO2' with several Ruby files. The code editor displays the content of 'ejercicio1.rb', which defines a hash of contacts and a method to look up a phone number by name. The terminal shows the execution of the program, displaying the list of contacts and the result of a lookup for 'Maria'.

```
ejercicio1.rb
1  contactos = {
2    'Maria' => '55026619',
3    'Pedro' => '89524437',
4    'Juan' => '86053475',
5    'Alberto' => '83389539'
6  }
7
8  puts "Listado de contactos:"
9  contactos.each do |nombre, celular|
10   puts "#{nombre}: #{celular}"
11 end
12
13 print "Por favor introduce el nombre para buscar el número de celular: "
14 nombre_buscar = gets.chomp
```

```
root@debian:/home/joyasi/trabajo2# ruby ejercicio1.rb
Listado de contactos:
Maria: 55026619
Pedro: 89524437
Juan: 86053475
Alberto: 83389539
Por favor introduce el nombre para buscar el número de celular: Maria
El número de celular de Maria es: 55026619
root@debian:/home/joyasi/trabajo2#
```

11. 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.



The screenshot shows the Visual Studio Code interface with a Ruby file named `ejerciciop2.rb` open. The code defines a `Calcular` class with two methods: `suma_pares` and `suma_impares`. The program prompts the user to enter a number, calculates the sum of even and odd digits, and prints the results.

```
1 class Calcular
2   def suma_pares
3     # ... (code for summing even digits) ...
4   end
5   def suma_impares
6     # ... (code for summing odd digits) ...
7   end
8 end
9
10 print "Introduce un número: "
11 numero = gets.chomp
12
13 calculadora = Calcular.new(numero)
14
15 puts "La suma de los números pares en #{numero} es: #{calculadora.suma_pares}"
16 puts "La suma de los números impares en #{numero} es: #{calculadora.suma_impares}"
```

The terminal output shows the program being executed with the input `23`:

```
root@debian:/home/joysi/trabajo2# ruby ejerciciop2.rb
Introduce un número: 23
La suma de los números pares en 23 es: 2
La suma de los números impares en 23 es: 3
root@debian:/home/joysi/trabajo2#
```

12. Extra: hacer que el programa número 1 que se ejecute varias veces hasta presionar una tecla.

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files under 'TRABAJO2', including 'ejerciciop1.rb' through 'ejerciciop9.rb'. The file 'ejerciciop3.rb' is selected. The main editor displays the code for 'ejerciciop3.rb', which defines a hash of contacts and a method to list them. The terminal at the bottom shows the command 'ruby ejerciciop3.rb' being executed, followed by the program's output: a list of contacts and a prompt to search for a contact by name. The user has entered 'maria'.

```
ejerciciop3.rb
1 contactos = {
2   'Maria' => '55026619',
3   'Pedro' => '89524437',
4   'Juan' => '86053475',
5   'Alberto' => '83389539',
6 }
7 def mostrar_contactos(contactos)
8   puts "Listado de contactos:"
9   contactos.each do |nombre, celular|
10    puts "#{nombre}: #{celular}"
11  end
12 end
13 loop do
14   mostrar_contactos(contactos)
```

```
root@debian:/home/joysi/trabajo2# ruby ejerciciop3.rb
Listado de contactos:
Maria: 55026619
Pedro: 89524437
Juan: 86053475
Alberto: 83389539
Introduce el nombre para buscar el número de celular (o escribe 'salir' para terminar): mar
ia
El número de celular de Maria es: 55026619
Listado de contactos:
Maria: 55026619
Pedro: 89524437
Juan: 86053475
Alberto: 83389539
Introduce el nombre para buscar el número de celular (o escribe 'salir' para terminar):
```

12

This screenshot shows the terminal window from the previous image, continuing the execution of the Ruby program. The user has entered 'pedro' in response to the search prompt. The program outputs the contact information for Pedro and then lists all contacts again. The terminal text is as follows:

```
El número de celular de Maria es: 55026619
Listado de contactos:
Maria: 55026619
Pedro: 89524437
Juan: 86053475
Alberto: 83389539
Introduce el nombre para buscar el número de celular (o escribe 'salir' para terminar): ped
ro
El número de celular de Pedro es: 89524437
Listado de contactos:
Maria: 55026619
Pedro: 89524437
Juan: 86053475
Alberto: 83389539
Introduce el nombre para buscar el número de celular (o escribe 'salir' para t
```

