INSTITUTO FEDERAL
São Paulo
Câmpus Cubatão

Alexandre Roberto Anjos dos Santos

Ana Beatriz Rodrigues Mesquita

Felipe Fernandes Valente

Gabriel Cruz do Carmo

Gabriel Vinícius de Moraes Conceição

Gabriella de Sousa

Giovana Vieira dos Santos

Gustavo Henrique Gesualdo dos Santos

Luana Tomaz Accorsi

Luisa Carvalho Souza de Oliveira

Maicon André Lima Santos

CTII 417

RUBY

Cubatão

2021

Alexandre Roberto Anjos dos Santos

Ana Beatriz Rodrigues Mesquita

Felipe Fernandes Valente

Gabriel Cruz do Carmo

Gabriel Vinícius de Moraes Conceição

Gabriella de Sousa

Giovana Vieira dos Santos

Gustavo Henrique Gesualdo dos Santos

Luana Tomaz Accorsi

Luisa Carvalho Souza de Oliveira

Maicon Andre Lima Santos

CTII 417

RUBY

Trabalho de Conclusão de Curso, apresentado ao curso de Informática, do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo – Campus Cubatão como requisito parcial para obtenção do título Técnico em Informática.

Orientador: Prof. Mauricio Neves Asenjo.

Cubatão

2021

SUMÁRIO

1. CAPÍTULO: HISTÓRICO E IDE´S

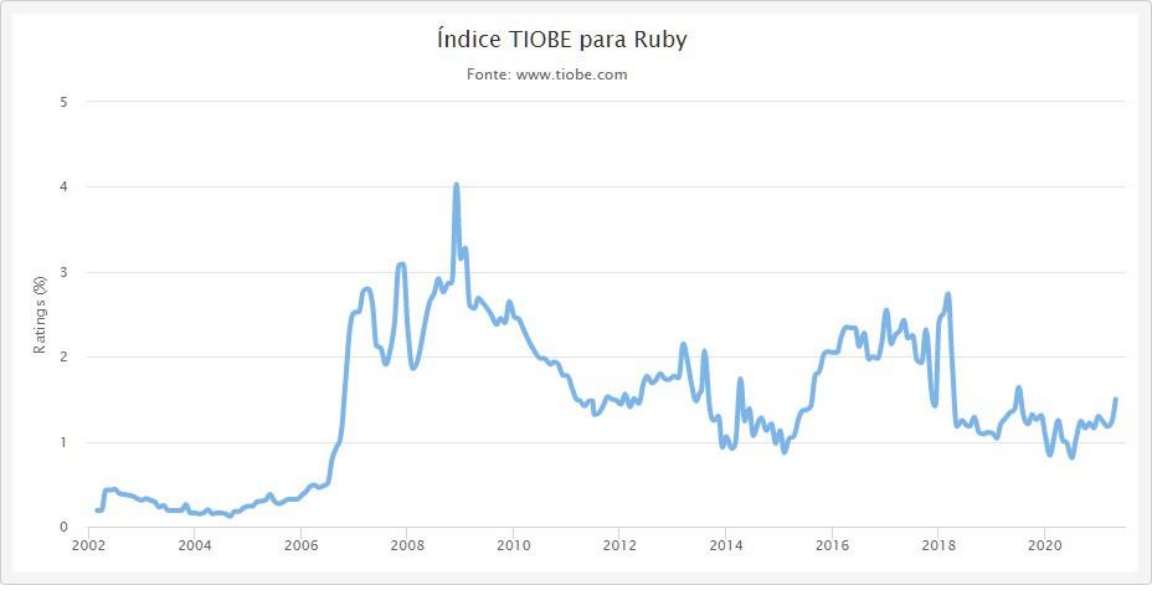
1.1. CRIAÇÃO E CARACTERÍSTICAS DEDESENVOLVIMENTO

Tratando-se do desenvolvimento dessa linguagem, Yukihiro Matz Matsumoto pode levar o crédito: ele a desenvolveu por volta da década de 90, inspirando-se em partes de outras linguagens, tais como, por exemplo, SmallTalk, Ada e Lisp, com o intuito de uma linguagem script que fosse mais poderosa do que a Perl e mais orientada a objetos do que a Phyton.Ruby é uma linguagem de programação interpretada multiparadigma, de tipagem dinâmica e forte, com gerenciamento de memória automático, para ser usada como linguagem de script.

Para aqueles que buscam algo dentro do mercado startup, a linguagem Ruby é o ponto ideal para busca de conhecimento. Ruby está presente em diversos serviços pelo mundo, muitos deles fortemente utilizados como o Twitter. Sua característica principal é de sintaxe com uma leitura simples que permite ao usuário utilizar pouco código, mas de uma forma que suas aplicações funcionem.

A linguagem Ruby é open source e conta com uma grande comunidade na área de desenvolvimento, se mostrando sempre empenhada para atualizações e melhorias dentro do seu funcionamento e aprimoramento da linguagem.

De acordo com o índice TIOBE (o qual avalia a popularidade das linguagens de programação por meio de mecanismos de pesquisa), em maio de 2021, RUBY encontrou-se em 11º lugar, sendo sucedido por Classic Visual Basic e antecedido



por SQL.

Por meio do gráfico acima se compreende que a popularidade da linguagem permeia o 1% desde maio de 2018, com poucos aumentos consideráveis. No entanto, um dos motivos dela ainda permanecer entre as favoritas é a questão de quefoi criada para ser uma linguagem mais fácil de compreender e mais agradável de programar, outro motivo é que ela seria um forte na linguagem funcional, ou seja, tem recursos poderosos e essenciais.

1.2. IDE's

1.2.1. PRINCIPAIS IDE´S

Ruby é uma linguagem muito difícil de usar IDE's, e muitos optam por usarem editores de texto, como por exemplo, o GEdit. Mas ainda assim, muitas pessoas usufruem de IDE's, entre as principais estão:

- **RubyMine IDE:** uma IDE fácil de se utilizar, desenvolvido pela JetBrains. Ele oferece suporte completo para linguagens como CoffeeScript, desenvolvimento Ruby on Rails, ERB, JavaScript, CSS, HAML e Saas, porém é uma plataforma não gratuita;
- **Aptana Studio:** considerado o melhor IDE para Ruby on Rails, pois fornece suporte estendido para várias linguagens de programação, incluindo AJAX, Javascript, Ruby on Rails, CSS, PHP e HTML, é um dos IDEs de software livre mais favorecidos que ajuda a criar aplicativos da web dinâmicos e de programação;
- **Selenium:** uma estrutura de teste de software, que vem com um punhado de recursos, como preenchimento automático, teste, depuração e ferramentas de gravação, suporta também desenvolvimento Python, .NET, Perl e JAVA;
- **Atom Editor:** suporta várias linguagens de programação que incluem Ruby, é extremamente personalizável para simplificar o desenvolvimento de código, contudo ele não funciona sozinho, pois aproveita o suporte da prática integrada do Github;
- **Emacs:** desenvolvido por Richard Stallman, Emacs é um editor completo e adorado por vários desenvolvedores Ruby, pois é completo para scripts Ruby e programação web baseada em Ruby on Rails, sendo também totalmente gratuito;
- **Netbeans:** desenvolvido pela Oracle, o Netbeansé totalmente gratuito e utilizado para diversas linguagens de programação como um IDE predominante para o

desenvolvimentode linguagens como: Java, PHP, HTML 5, C ou C ++. É considerada uma forma mais rápida e inteligente de codificar e define a referência para o desenvolvimento de aplicativo com tecnologias de ponta, sem mencionar que contém um excelente suporte para tecnologias populares, como Maven.

1.3. ATOM

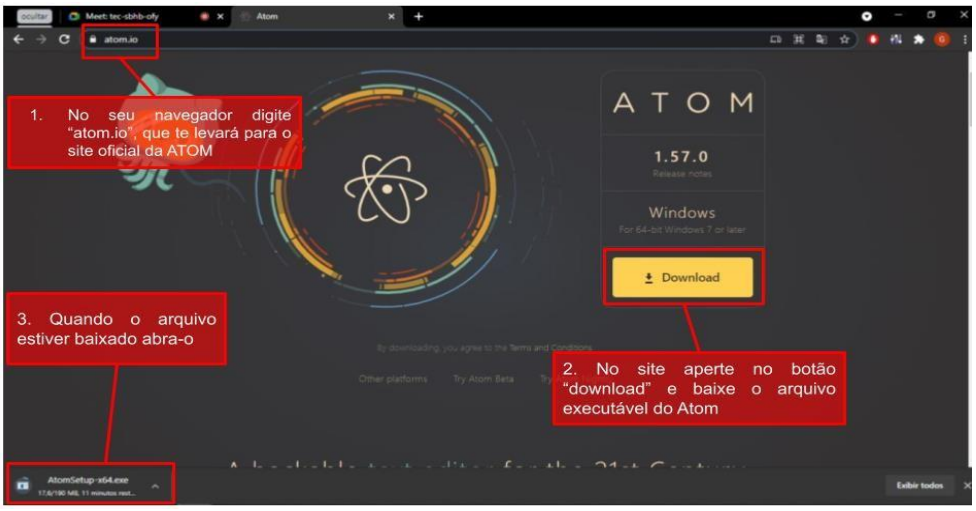
Inicialmente, selecionou-se como plataforma de desenvolvimento o NetBeans IDE, pela experiência e familiaridade que acerca do ambiente de desenvolvimento, entretanto, a versão do NetBeans que compactua com a linguagem Ruby, não está mais disponível atualmente. Diante disso, optou-se por escolher o Atom IDE.

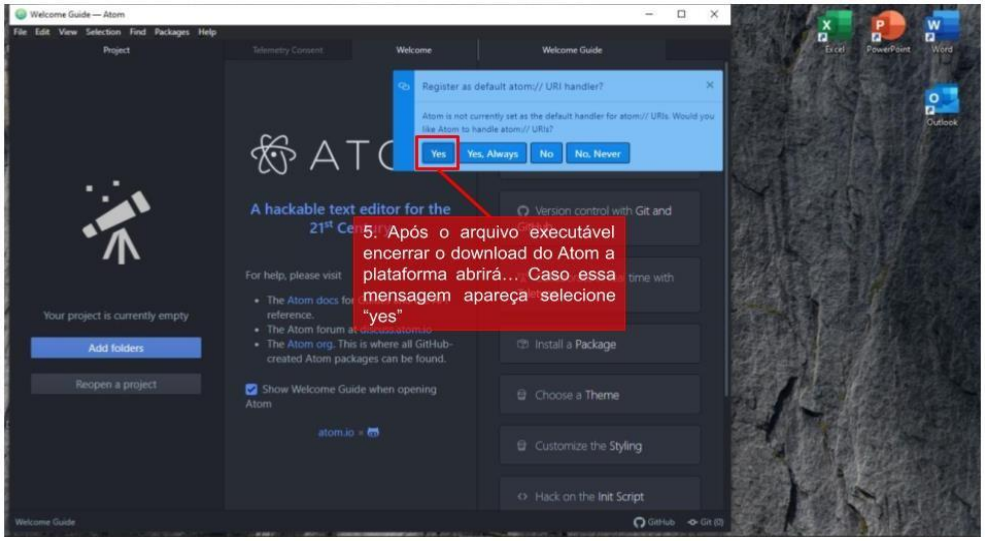
A preferência pelo Atom IDE deteve como principal fator as críticas positivas sobre a plataforma de desenvolvimento que é constituída por diversas vantagens.Alémde o Atom ser umas das poucas IDE’s gratuitas e de qualidade para desenvolvedores de software, outro critério para a seleção foi a facilidade de instalação e utilização da plataforma, que oferece todas as ferramentas necessárias para a criação de programas mais simples até os mais profissionais, sem impossibilitar o acesso ao grande conjunto de bibliotecas, rotinas, módulos e ferramentas para a criação de programas.

A plataforma possui um sistema de controle que permite trabalhar com várias versões de arquivos inseridos em um diretório, localizados localmente ou remotamente, preservando suas versões antigas e informações de quem e quando manipulou os arquivos. Além disso, o Atom, ao contrário de algumas outras IDE’s, possibilita alterar o idioma do ambiente, facilitando o acesso e auxiliando no desenvolvimento do programa de maneira mais rápida.

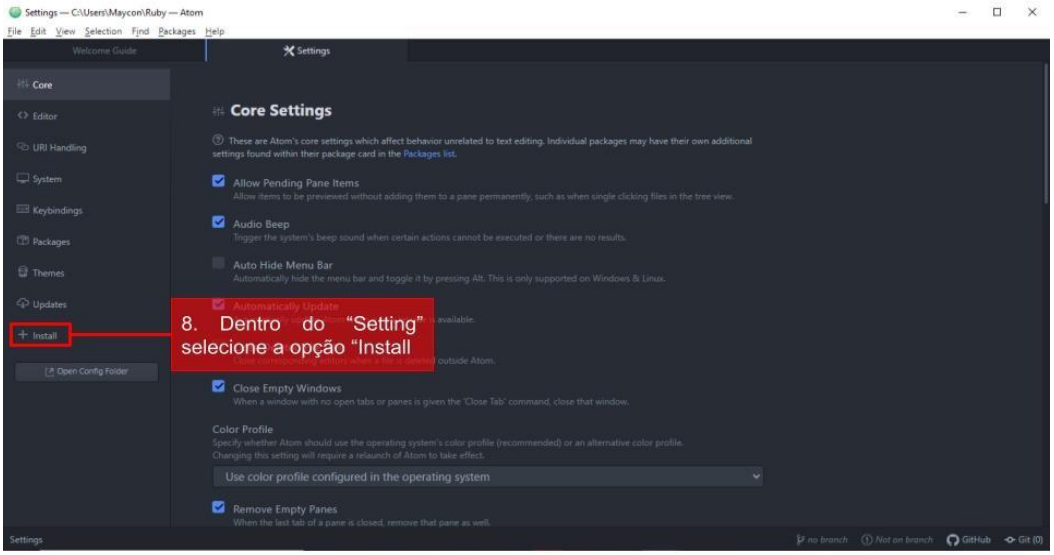
1.3.1. INSTALAÇÃO

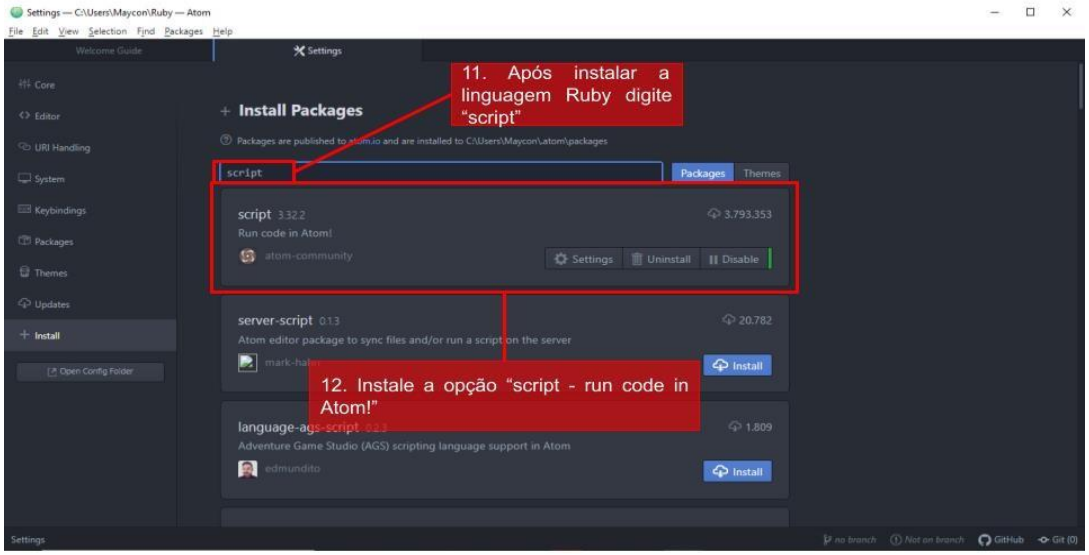
Segue abaixo o tutorial de instalação da IDE Atom e a configuração em relação à linguagem Ruby:

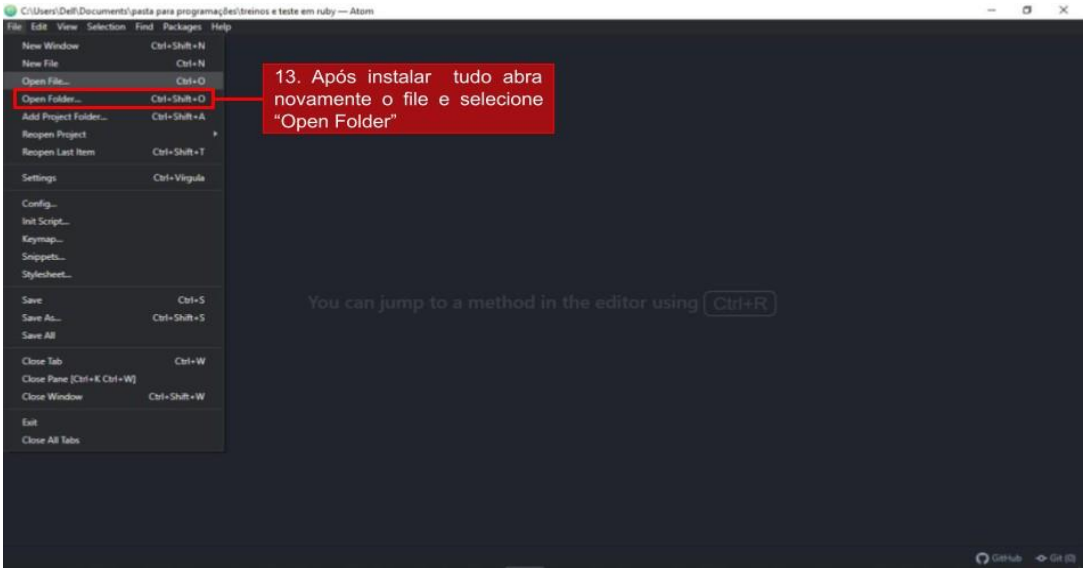


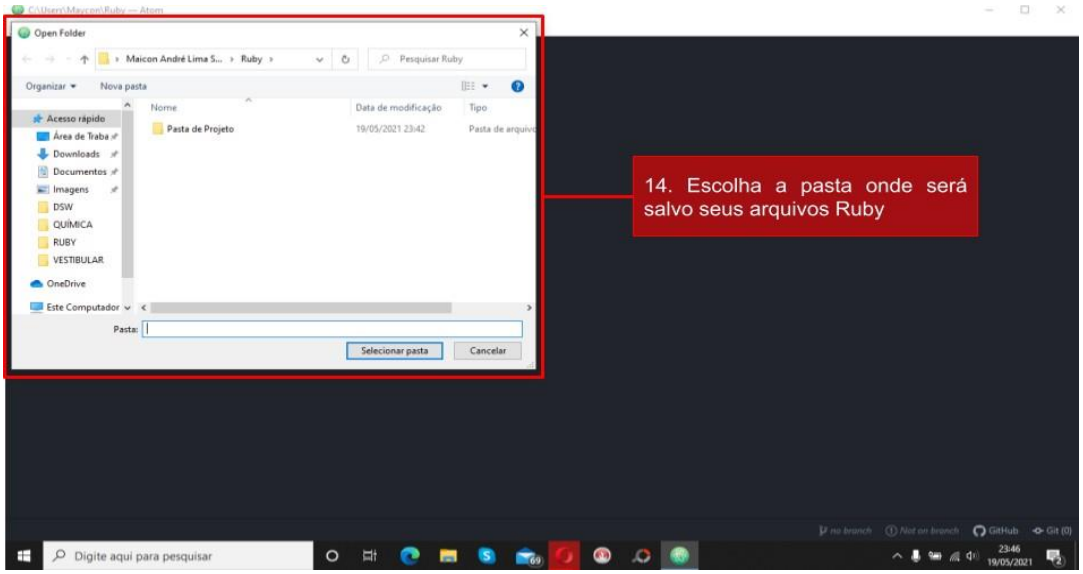


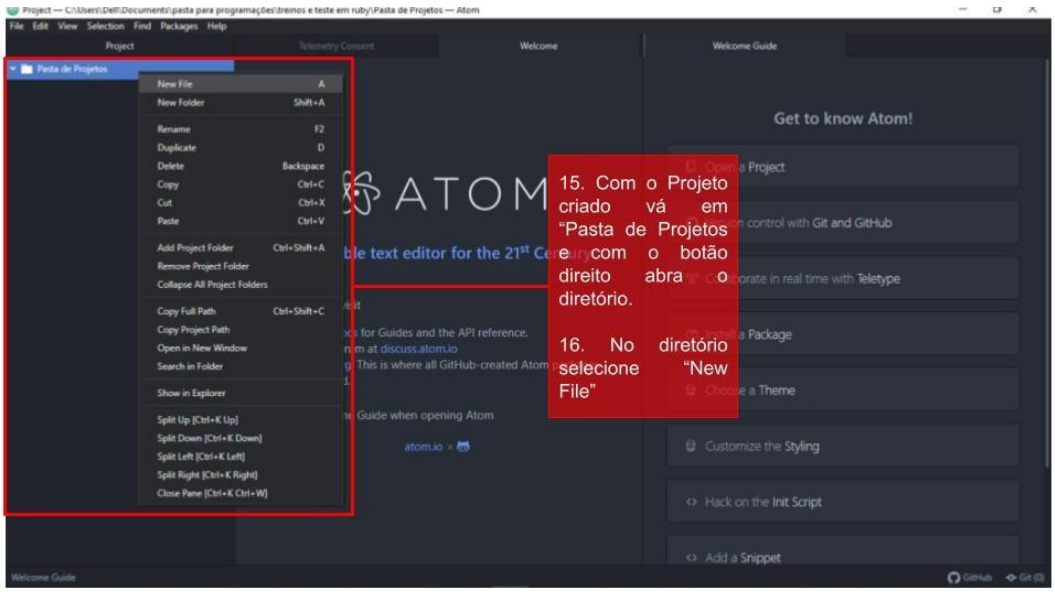

6. Dentro da IDE, abra o diretório “File
7. Abra o “setting”
You can jump to a method in the editor using Ctrl+R

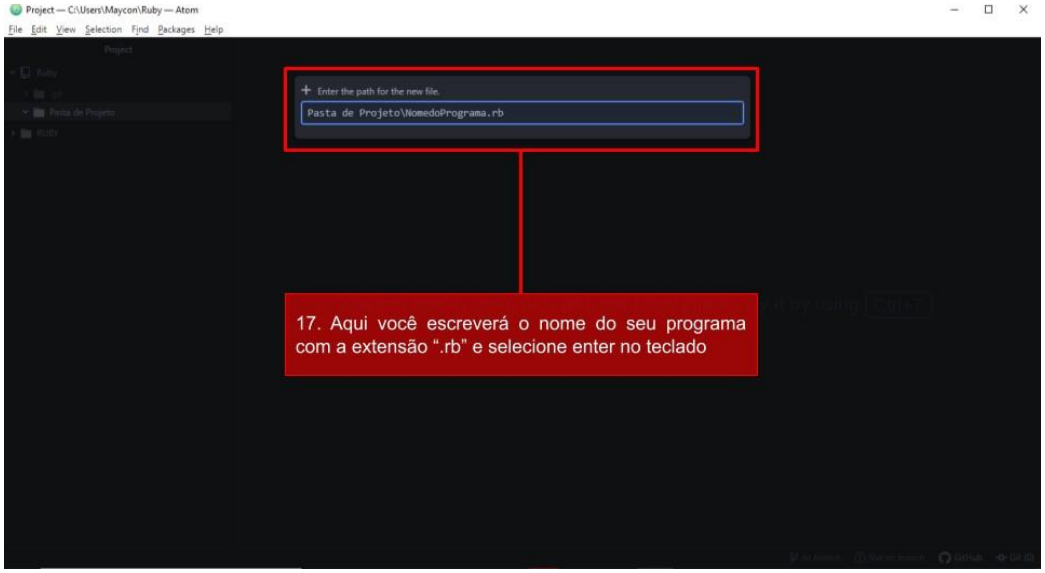

8. Dentro do “Setting” selecione a opção “Install
Core Settings

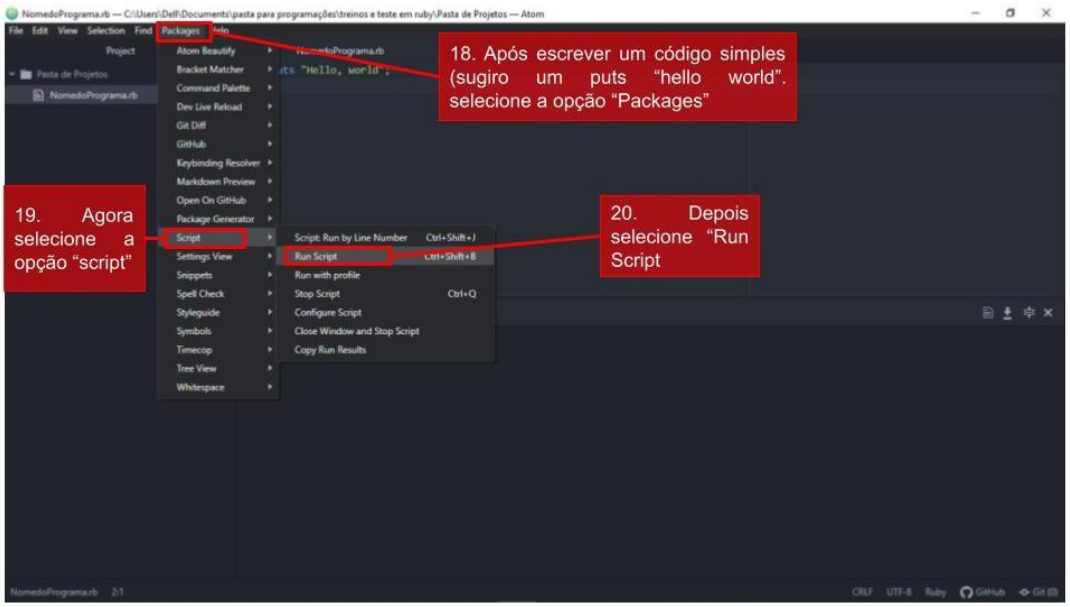

11. Após instalar a linguagem Ruby digite “script”
12. Instale a opção “script - run code in Atom!”
Install Packages
script
Packages
Themes

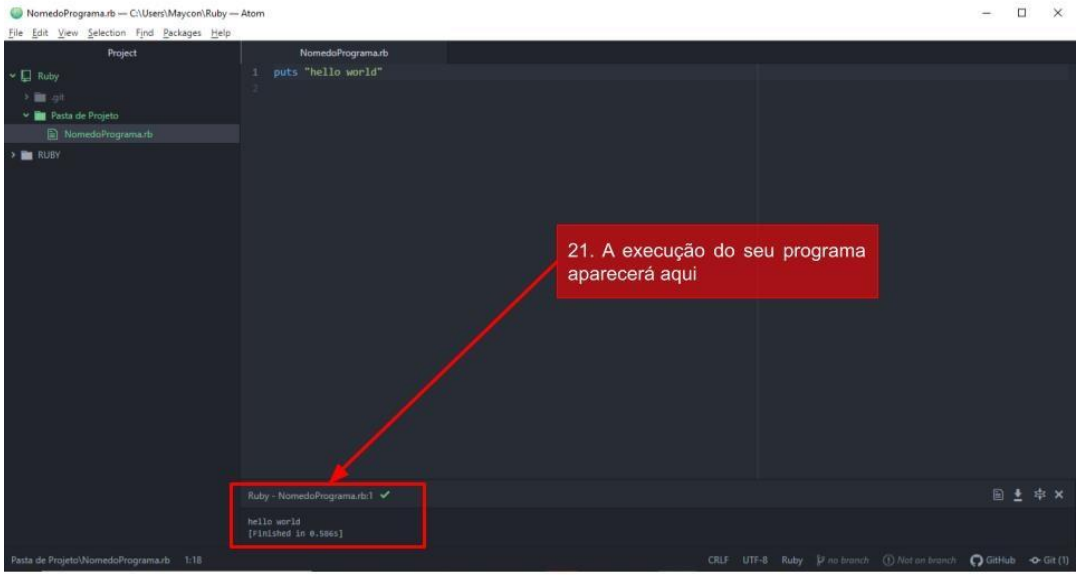

13. Após instalar tudo abra novamente o file e selecione "Open Folder"


14. Escolha a pasta onde será salvo seus arquivos Ruby


15. Com o Projeto criado vá em "Pasta de Projetos e com o botão direito abra o diretório.
16. No diretório selecione "New File"


Project — C:\Users\Maycon\Ruby — Atom
File Edit View Selection Find Packages Help
Enter the path for the new file.
Pasta de Projeto\NomedoPrograma.rb
17. Aqui você escreverá o nome do seu programa com a extensão ".rb" e selecione enter no teclado


18. Após escrever um código simples (sugiro um puts "hello world". selecione a opção "Packages"
19. Agora selecione a opção "script"
20. Depois selecione "Run Script
Packages
Script
Run Script
Ctrl+Shift+B


NomedoPrograma.rb — C:\Users\Maycon\Ruby — Atom
puts "hello world"
21. A execução do seu programa aparecerá aqui
Ruby - NomedoPrograma.rb:1
hello world
[Finished in 0.586s]

1.3.2. ATOM: "HELLO WORLD"

Tal como a tradição iniciada pelos americanos Brian Kernighan e Dennis Ritchie (criadores da linguagem de programação C) que, no livro de 1978, intitulado “The C Programming Language”, utilizaram pela primeira vez a frase “Hello World!”, repetiu-se essa clássica mensagem em Ruby, para tal, digitou-se: puts(“Hello, world”), comando o qual exibe na tela o que está escrito entre parênteses e aspas dupla, imprimindo-o no console.

Note que no canto inferior aparece a confirmação da execução bem sucedida e há a impressão no console.

2. CAPÍTULO: LAÇOS DE REPETIÇÃO E RUBY NA PRÁTICA

2.1. EXECUTANDO PROGRAMAS: COMANDOSINICIAIS, OPERADORES ARITMÉTICOS, FUNÇÕES MATEMÁTICAS E DECISÃO LÓGICA.

Na programação o que mais se utiliza são os operadores aritméticos, eles são responsáveis pelas funções mais simples. No mundo real, utilizamo-nos sempre: “+”,”-“,”*”,”%” e “/”-os apresentados são operadores aritméticos-; mas na área de codificação há mais tipos de operadores(relacionais- “==”; “>=”; “<=”; “>”; “<”;” <=>”; “===”; “.eql?”; “'Igual?'" -, lógicos –“&&”; “||”; “!”-, de atribuição(possuem a mesma lógica que o C#) – “: =”; “+=”; “-=”; “*=”;” /=”; “%=”;” **=”- e os de alcance-“(..)”; “(...)”.

É interessante mencionar que o “===” testa a igualdade na instrução maiúscula e minúscula. O “.eql?” retorna verdadeiro se as variáveis forem iguais no tipo e nos valores. Por fim, o “.Igual?” retorna verdadeiro se elas tiverem o mesmo id do objeto. Além disso, o operador “**” não indicia multiplicação/ e, sim,exponenciação entre números.

Vale ressaltar que o operador “<=>” é um “coringa”, pois ,numa comparação entre dois valores, ele retorna 0 se os números forem iguais ; 1 se o primeiro for maior; e -1 se o primeiro for menor.

```
if 5<=> 6
#vai retornar -1
 if 5<=>5
#vai retornar 0
 if 5<=>4
#vai retornar 1
```

Pode-se notar que a estrutura IF-ELSE(análise de condição) de Ruby é bem parecida com a da linguagem C#, a única diferença é que se há muitos códigos dentro de uma decisão, o C# utiliza de “{}” para indicar início e fim; já a Ruby utiliza de “end”.

Ademais, o operador “..”,não existe em C#, é usado para criar intervalos; masé diferente do “...”, como demonstrado no exemplo abaixo.

```
5..8
#é referente a:5,6,7,8.
5...8
#é referente a:5,6,7. Exclui-se o
último número da sequência.
```

2.1.1. ÁREA DE TRIÂNGULO

O código, a seguir,demonstra, por meio da linguagem Ruby como calcular a área de um triângulo. ATENÇÃO: o comando “gets....l” é usado para ler números inteiros, o “gets....s” para string e o “gets...f” para ler números do tipo float.

Areadotriangulo.rb

```ruby
puts "\nDigite o valor da base do triângulo"
STDOUT.flush
base = gets.chomp.to_i

puts "\nDigite o valor da altura do triângulo"
STDOUT.flush
altura = gets.chomp.to_i

Area = base * altura / 2

puts "A area do triângulo é: #{Area}"
```

2.1.2. SENO E COSSENO DE ÂNGULO

Este programa abaixo mostra como, de maneira simples, pode-se obter o seno e cosseno de ângulos. Note que o cálculo é feito a partir do valor em graus, se fosse calcular o seno/cosseno a partir do ângulo em radiano precisaria fazer outro cálculo.

Angulo.rb

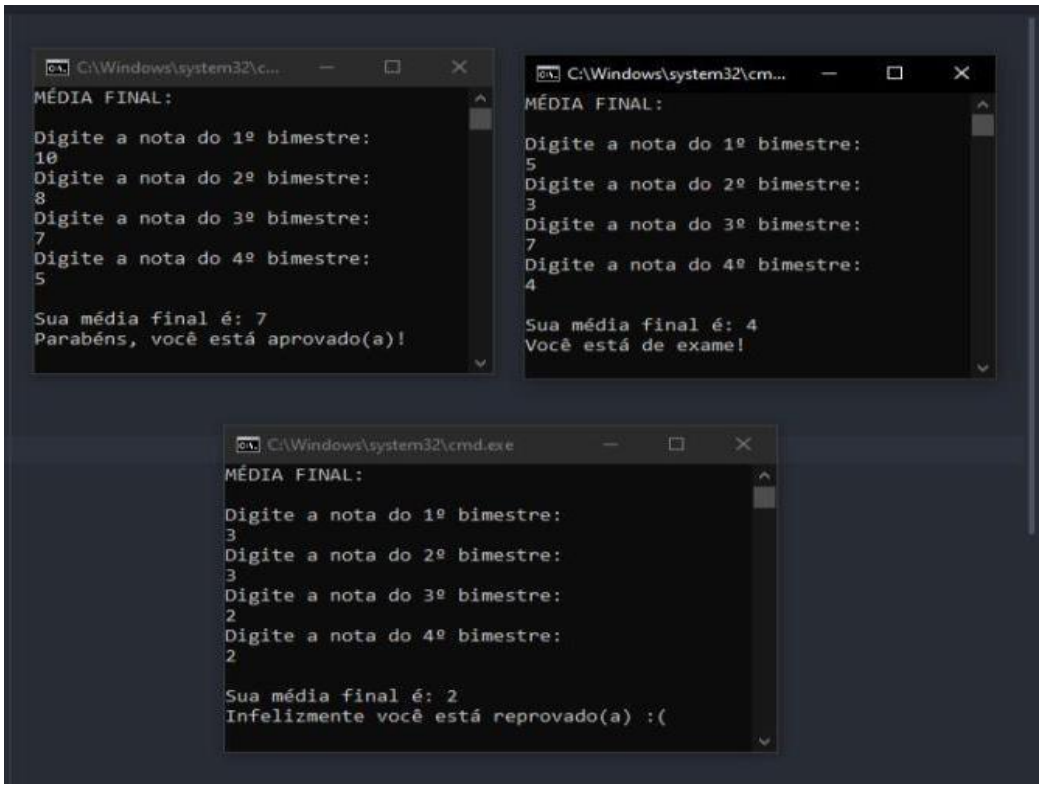
```ruby
puts "\nDigite o ângulo em graus"
STDOUT.flush
angulo = gets.chomp.to_f

a = (angulo * Math::PI)/180

print "cos: "

puts Math.cos(a)

print "sen: "

puts Math.sin(a)
```

2.1.3. MÉDIA DE NOTAS E APROVAÇÃO

Através do programa abaixo, é calculada a media aritmética dos valores inseridos pelo usuário, destacando, por meio do uso de if-else, se o detentor das notas digitadas foi aprovado ou reprovado.

```ruby
puts ""
puts "MÉDIA FINAL:"
puts ""

#o usuário digitará as suas respectivas notas de cada um dos 4 bimestres
#e o programa irá armazená-las; a função "gets.chomp.to_i" irá ler o valor
#do tipo inteiro.

puts "Digite a nota do 1º bimestre: "
n1 = gets.chomp.to_i

puts "Digite a nota do 2º bimestre: "
n2 = gets.chomp.to_i

puts "Digite a nota do 3º bimestre: "
n3 = gets.chomp.to_i

puts "Digite a nota do 4º bimestre: "
n4 = gets.chomp.to_i

#a váriavel media armazenará o calculo da média aritmetica das notas que
#estão nas variaveis n1, n2, n3 e n4 e será apresentado para o usuário.

media = (n1 + n2 + n3 + n4)/4
puts ""
puts "Sua média final é: #{media}"

#impõe-se uma condição if e else para saber se o aluno está aprovado,
#reprovado ou de exame de acordo com o seguinte critério:
```

```ruby
puts "Digite a nota do 3º bimestre: "
n3 = gets.chomp.to_i

puts "Digite a nota do 4º bimestre: "
n4 = gets.chomp.to_i

#a váriavel media armazenará o calculo da média aritmetica das notas que
#estão nas variaveis n1, n2, n3 e n4 e será apresentado para o usuário.

media = (n1 + n2 + n3 + n4)/4
puts ""
puts "Sua média final é: #{media}"

#impõe-se uma condição if e else para saber se o aluno está aprovado,
#reprovado ou de exame de acordo com o seguinte critério:
#Média final maior ou igual a seis o aluno esta aprovado;
#Entre 3 e 6 de exame;
#Menor que três reprovado,

if media >= 6
    puts"Parabéns, você está aprovado(a)!"
else
  if media >= 3 && media < 6
      puts"Você está de exame!"
else
    puts"Infelizmente você está reprovado(a) :("
end
end
```



2.1.4. OPERAÇÕES BÁSICAS ENTRE DOIS NÚMEROS: IF

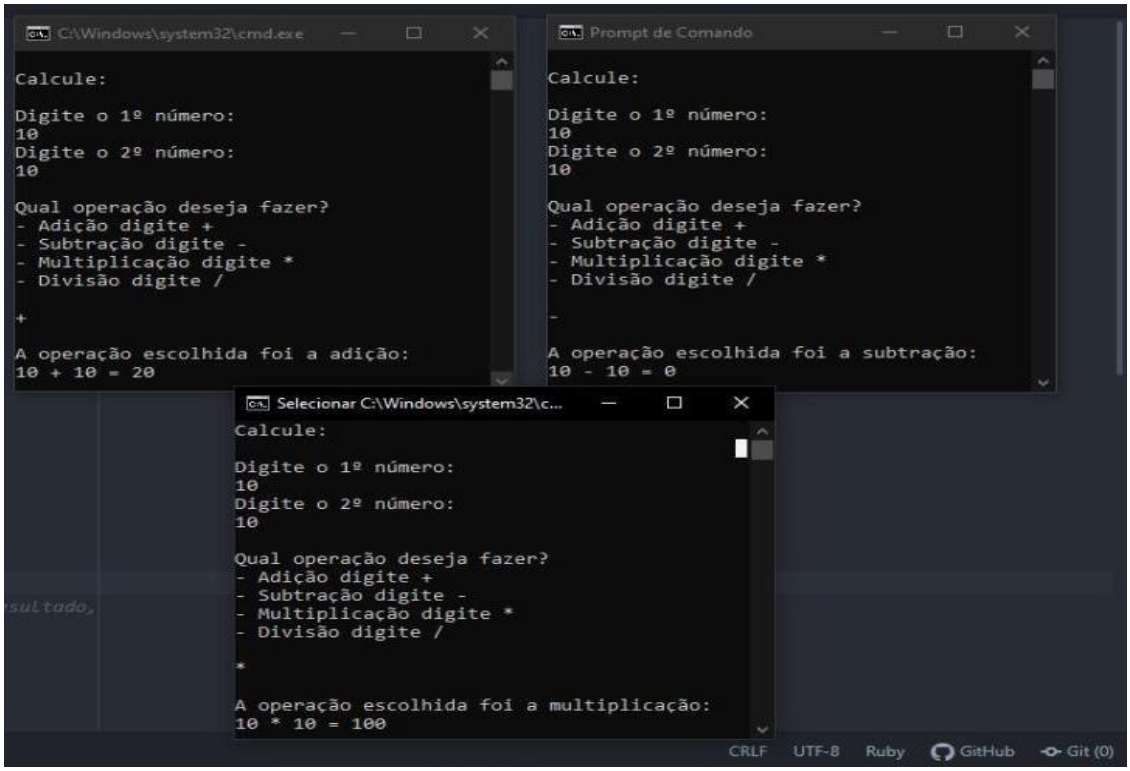
Neste código abaixo é exibido um menu para que o usuário escolha qual operação aritmética quer fazer com valores, os quais digitou anteriormente.

```ruby
puts ""
puts "Calcule:"
puts ""

#O usuário fará a digitação dos dois números a
#serem calculados, após a digitação esses números serão
#armazenados em variaveis
puts "Digite o 1º número: "
num1 = gets.chomp.to_i

puts "Digite o 2º número: "
num2 = gets.chomp.to_i

#O usuário irá digitar a operação que deseja fazer e esse carectere
#ficará armzenado na variavel op
puts ""
puts "Qual operação deseja fazer?"
puts "- Adição digite +"
puts "- Subtração digite -"
puts "- Multiplicação digite *"
puts "- Divisão digite /"
puts""
op = gets.chomp.to_s

#Condições de if e else para fazer o respectivo cálculo e exibição do res
#cujo caractere/operação digitado/a corresponde.
if op == "+"
puts""
puts "A operação escolhida foi a adição: "
puts "#{num1} + #{num2} = #{num1 + num2}"
```

doisnumif.rb 24:1

```ruby
if op == "+"
puts""
puts "A operação escolhida foi a adição: "
puts "#{num1} + #{num2} = #{num1 + num2}"
else

  if op == "-"
  puts""
  puts "A operação escolhida foi a subtração: "
  puts "#{num1} - #{num2} = #{num1 - num2}"
else

  if op == "*"
  puts""
  puts "A operação escolhida foi a multiplicação: "
  puts "#{num1} * #{num2} = #{num1 * num2}"
else

  if op == "/"
  puts""
  puts "A operação escolhida foi a divisão: "
  puts "#{num1} / #{num2} = #{num1 / num2}"

else
  puts""
  puts"Não temos essa operação!!"
end
end
end
end
```

doisnumif.rb 31:1



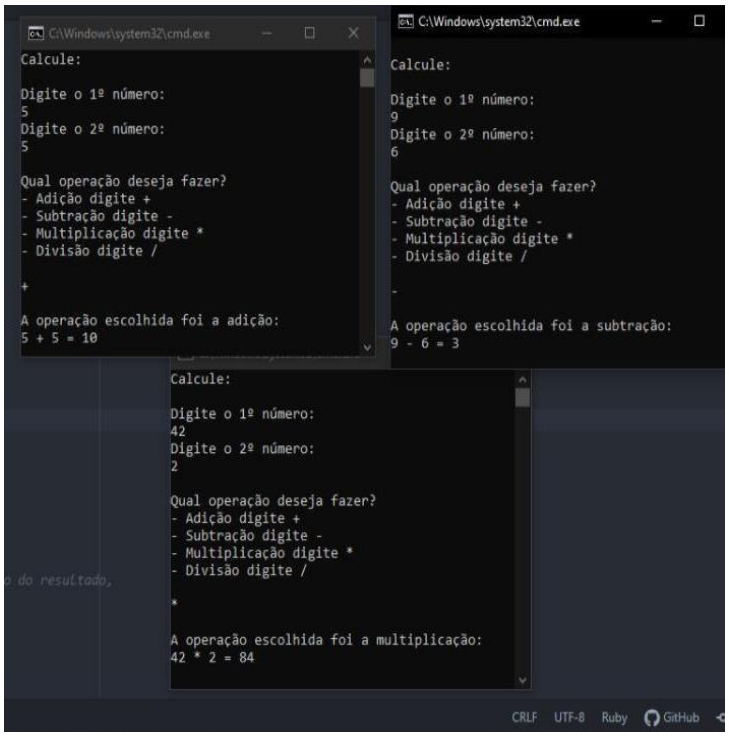
2.1.5. OPERAÇÕES BÁSICAS ENTRE DOIS NÚMEROS: SWITCH-CASE

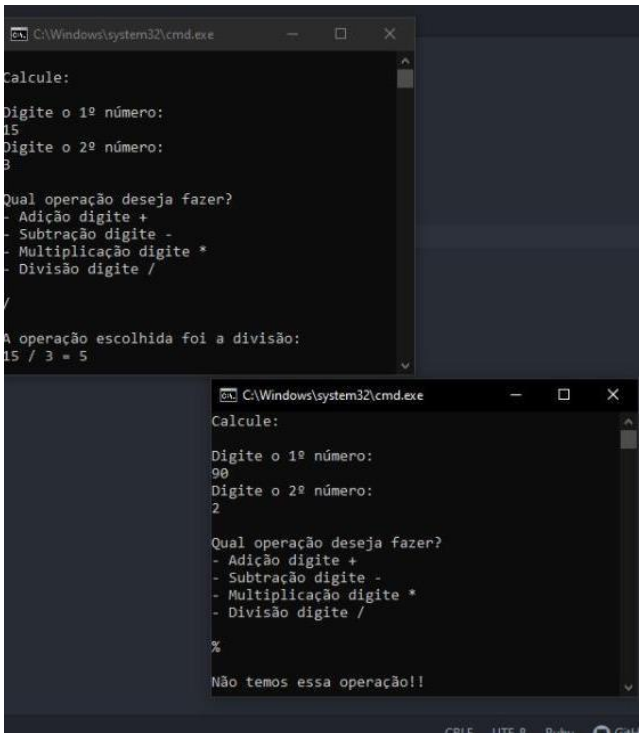
No Ruby, a estrutura SWITCH-CASE, encontrado no C#, é tratado como CASE-WHEN. Nesse sentido, possui a mesma lógica que o C# mas com “nome” diferente. No código abaixo a exemplificação de seu uso a partir de uma consulta de operações matemáticas.

```ruby
puts ""
puts "Calcule:"
puts ""

#O usuário fará a digitação dos dois números a
#serem calculados, após a digitação esses números serão
#armazenados em variaveis
puts "Digite o 1º número: "
num1 = gets.chomp.to_i

puts "Digite o 2º número: "
num2 = gets.chomp.to_i

#O usuário irá digitar a operação que deseja fazer e esse carectere
#ficará armzenado na variavel op
puts ""
puts "Qual operação deseja fazer?"
puts "- Adição digite +"
puts "- Subtração digite -"
puts "- Multiplicação digite *"
puts "- Divisão digite /"
puts""
op = gets.chomp.to_s

#Condições no Switch/Case para fazer o respectivo cálculo e exibição
#cujo caractere/operação digitado/a corresponde.
case op
when "+"
  puts""
  puts "A operação escolhida foi a adição: "
```

doisnumswitch.rb 18:25

```ruby
puts "- Multiplicação digite *"
puts "- Divisão digite /"
puts""
op = gets.chomp.to_s

#Condições no Switch/Case para fazer o respectivo cálculo e exibição do resultado,
#cujo caractere/operação digitado/a corresponde.
case op
when "+"
  puts""
  puts "A operação escolhida foi a adição: "
  puts "#{num1} + #{num2} = #{num1 + num2}"
when "-"
  puts""
  puts "A operação escolhida foi a subtração: "
  puts "#{num1} - #{num2} = #{num1 - num2}"
when "*"
  puts""
  puts "A operação escolhida foi a multiplicação: "
  puts "#{num1} * #{num2} = #{num1 * num2}"
when "/"
  puts""
  puts "A operação escolhida foi a divisão: "
  puts "#{num1} / #{num2} = #{num1 / num2}"
else
  puts""
  puts"Não temos essa operação!!"
end
```



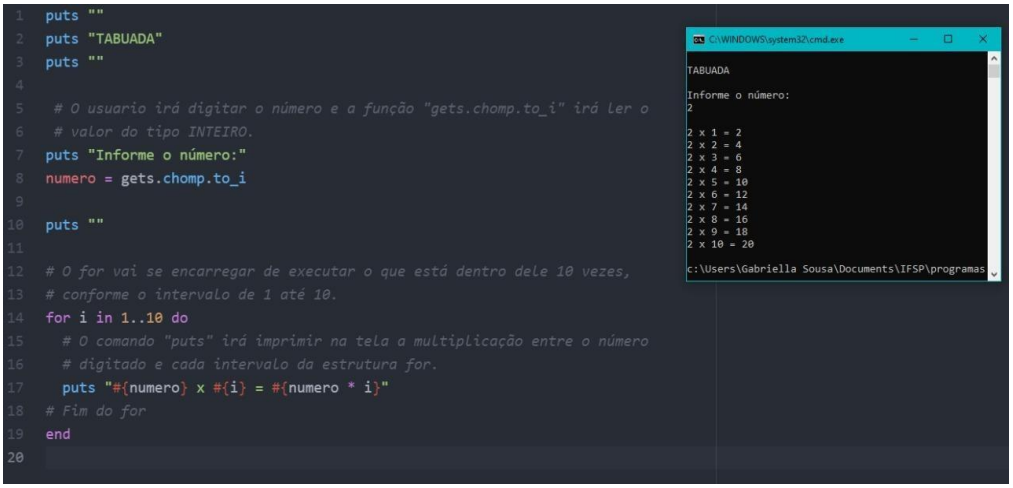


2.2. ARRAYS E REPETIÇÃO DE CÓDIGO

Há momentos em que variáveis não são eficazes e nem funcionais para o objetivo de determinados programas. Assim, utiliza-se os arrays(vetores), os quais conseguem guardar inúmeros valores; ao contrário da variável que só guarda um por vez. Ademais, diversas aplicações no cotidiano necessitam que o software rode mais do que uma vez. Desse modo, existem os laços de repetições que, como o próprio nome diz, permite a repetição dos comandos determinadas vezes até que encontre o ponto ‘break”/”end”. Note que a estrutura FOR de Ruby é um pouco diferente do C#.

2.2.1. TABUADA

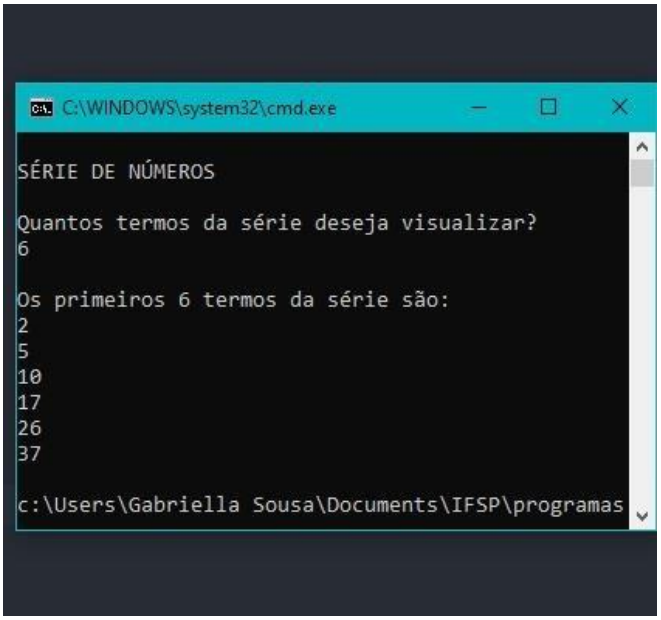
Aqui, o usuário digita um número qualquer e exibe-se a tabuada dele.



2.2.2. EXIBIÇÃO DE ‘N’ NÚMEROS DE UMA SÉRIE

De acordo com o valor digitado pelo usuário, exibe-se uma série de valores.

```ruby
puts ""
puts "SÉRIE DE NÚMEROS"
puts ""

# O usuario irá digitar a quantidade de termos e a função "gets.chomp.to_i"
# irá ler o valor do tipo INTEIRO.
puts "Quantos termos da série deseja visualizar?"
qtd_termo = gets.chomp.to_i

# A váriavel x irá receber a quantidade digitada + 1, para que o contador
# possa contar corretamente a sequência
x = qtd_termo + 1

puts ""

puts "Os primeiros #{qtd_termo} termos da série são: "

# O for irá executar do valor 1 até a quantidade digitada pelo usuário
for i in 1...x do
  # O comando "puts" irá imprimir na tela os termos da série digitados pelo
  # usuário, no qual o i irá variar entre 1 e a quantidade solicitada.
  puts "#{i**2 + 1}"
  #Fim do for
end
```



2.2.3. LISTAR NÚMEROS DA SEQUÊNCIA DE FIBONACCI MENORES QUE 1000

Com este código o usuário pode saber quais são os números que pertencem à sequencia de Fibonacci e que são menores que 1000.

```ruby
def numero_seguinte(n1,n2)
    n_seguinte=n1+n2
    return n_seguinte
end

sequencia = [0,1]
x = 2
n = 17

while x < n
    sequencia << numero_seguinte(sequencia[-1],sequencia[-2])
    x+=1

end

puts sequencia
```

2.2.4. COMPARAÇÃO ENTRE NÚMEROS

Neste próximo programa exemplifica-se como comparar dois números.

```ruby
# Digitação do primeiro número
puts 'Digite um valor:'
v1 = gets.to_i

# Digitação do segundo número
puts 'Digite um valor maior que o primeiro:'
v2 = gets.to_i

# Laço para que o segundo número seja maior que o primeiro
while v2 <= v1 do
    puts 'valor insuficiente, digite novamente'
    v2 = gets.to_i
end

#Prova de que o laço funciona
puts "#{v2} é maior que #{v1}... Obrigado!!!"
```

2.2.5. VALIDAÇÃO DE CARACTERE

Neste próximo programa exemplifica-se como validar caracteres.

```ruby
#Digitação do nome
puts 'Diga seu nome:'
nome = gets.chomp.to_s

#digitação do sexo
puts 'Seu sexo: [M – Masculino; F – Feminino]'
sexo =  gets.chomp.to_s

# Laço para que sexo só aceite "m" ou "f"
while sexo != 'm' and sexo != 'f' do
    puts 'Essa opção não existe, por favor seleciona uma das duas opções disponíveis'
    puts '[M – Masculino; F – Feminino]'
    sexo = gets.chomp.to_s
end

# Prova de que o laço funciona
if sexo == 'm'
    puts "Muito obrigado senhor #{nome}"
else
    puts "Muito obrigado senhora #{nome}"
end
```

2.2.6. FATORIAL DE NÚMEROS

Após a explicação da validação de um caractere, item anterior, é possível fazer uma rotina seguindo o mesmo raciocínio, porém com outro intuito: este programa irá calcular o fatorial de números e irá questionar o usuário se esse deseja repetir a ação.

Desse modo, nesse programa, pede-se o número, valida-o, faz-se o cálculo do fatorial, exibe o fatorial e, por fim, há a rotina para verificar se o usuário deseja calcular outro número ou se ele já finalizará o procedimento. Note que o comando “while” faz com que a ação seja repetida até a condição for verdadeira.

```ruby
keyf= true
chave= true
while keyf
puts "Digite um valor positivo e inteiro"
numero = gets.to_i
while numero < 0 do
  puts "Ooops! Tente novamente!"
  numero = gets.to_i
end
puts "O numero escolhido é o #{numero}"

# Função recursiva do fatorial
def fatorial(numero)
  if(numero > 1)
      return numero * fatorial(numero-1)
  else
      return 1
  end
end

# Calcula o fatorial
fat = fatorial(numero)

#Converte os números em texto e imprime o fatorial do número
puts "Fatorial de " + numero.to_s + ": " + fat.to_s

puts "Gostaria de realizar o calculo novamente? (1- SIM OU 2- NÃO)"
key = gets.to_i
while chave
if(key = 1)
```

```ruby
keyf=false
end
if (key = 2)
chave=false

else
  puts "digite um numero valido"
  key=gets.to_i
  chave= true
end

if(key == 1)
keyf=false
end
end
puts "Programa finalizado"
end
```

2.2.7. EXIBIÇÃO DE NÚMEROS NA ORDEM INVERSA À DIGITAÇÃO

O programa abaixo explica como armazenar números num Array(vetor) e exibir ao usuário na ordem inversa à digitação.

```ruby
# Criação do Vetor
num = []

# Inserindo os valores em cada parte do Vetor
for i in 1..10
    puts "Digite o #{i}º valor: "
    num[i] = gets.to_i
    # Essa variável auxiliar será usada abaixo
    aux = i
end

# Utilizando o aux imprima os valores de forma inversa
while aux >= 1 do
    puts "#{num[aux]}"
    aux = aux - 1
end
```

2.2.8. SOMATÓRIA E MÉDIA DE 10 VALORES

Sabendo armazenar numerais em um vetor é possível calcular a média desses. Abaixo temos um exemplo dessa aplicação utilizando como base 10 números.

```ruby
#criação do vetor e da variavel de soma
num = []
soma = 0

#Digitação dos 10 valores que serão armazenados vetor
for i in 1..10
    puts "digite o #{i}º número: "
    num[i] = gets.to_i

    #A variavel soma irá somar os valores que foram armazenados no vetor
    soma = soma + num[i]
end

#criação da variavel média como sendo o produto da divisão da variavel soma
media = soma / 10


#Mostrando os valores da soma e a media dos valores
puts "soma: #{soma}"
puts "média: #{media}"
```

2.2.9. ENCONTRAR MAIOR NÚMERO DE UMA SEQUÊNCIA

No programa abaixo, demonstra-se como calcular o maior número de uma sequência de 10 valores digitados pelo usuário.

```ruby
#declarando um Array

num = Array.new(10)

#estrutura de repetição para a inserção dos números

for fazer in 0..9
  x = fazer + 1
  print" Digite o #{x}º valor: "
  num[fazer]=gets.chomp.to_i
end

# declarando e dando valor as variaveis maior e menor

maior = num[0]
menor = num[0]

#verifcando qual numero é maior e menor

for menMai in 1..9
  if num[menMai] > maior
    maior = num[menMai]
  elsif num[menMai] < menor
    menor = num[menMai]
  end
end

#exibindo na tela os numeros.
puts"o maior número é o #{maior} e o menor número é o #{menor}"
```

2.2.10. CONSULTA DE NOMES

O código abaixo demonstra a possibilidade de criar uma agenda virtual, armazenando o nome e idade de uma pessoa.ATENÇÃO: o comando “loop” indica que o código vai ser executado até que uma condição seja verdadeira; ele só se encerra a partir do “break”(parar). Vale ressaltar que esse código poderia ter sido feito utilizando o Array bidimensional, o qual será explicado no tópico seguinte.

```ruby
#declaração dos arrays e das variáveis.
pessoa = Array.new(10)
idade = Array.new(10)
dnv = ''
busca =''
#estrutura de repetição para a digitação dos nomes e das idades.
for fazer in 0..9
    x = fazer + 1
    print" Digite o nome da #{x}º pessoa: "
    pessoa[fazer]=gets.chomp.to_str

    print" Digite a idade #{x}º pessoa: "
    idade[fazer]=gets.chomp.to_i
 end

#laço da pesquisa dos nomes.
loop do
#armazenando o nome da pessoa a ser pesquisada na variavel
    print" Digite o nome da pessoa: "
    busca = gets.chomp.to_str
    #laço para verificar se a pessoa existe ou não
    if(pessoa.include?(busca))
        for i in 0..9
            if (busca == pessoa[i])
        print "Nome: #{pessoa[i]}\nIdade: #{idade[i]}"
            end
        end
    else
    print"Pessoa não encontrada!"
    end

 #laço para verificar e o usuario gostaria de uma nova pesquisa ou não
    loop do
        print" \nDeseja fazer outra busca?S ou N "
        dnv = gets.chomp.to_str
        if (dnv == 'S' || dnv == 's' || dnv == 'n' || dnv == 'N')
          break
        else
            puts"Erro! Digite novamente"
        end

    end
    #condição para verificar se é necessário a volta do loop
    if (dnv == 'N' || dnv == 'n')
            break

    end

end
```

2.3. ARRAYS BIDIMENSIONAIS

2.3.1. OCUPAÇÃO DE CADEIRAS DE UM CINEMA

A partir da manipulação eficaz dos vetores unidimensionais, pode-se começar a trabalhar com os arrays bidimensionais: eles possuem linhas e colunas, portanto funcionam como uma tabela.

No programa abaixo, exemplifica-se como é feito a organização da ocupação de cadeiras de um cinema.

É importante ressaltar que nesse programa em especial, demonstra-se outra forma de fazer comentários ao decorrer do código.

```
=begin
  Declarando o array (Fazendo a Matriz)
  e uma variavel de auxilo
=end
num = Array.new(20){Array.new(15)}
a = 1

=begin
  inserindo valores na matriz para facilitar na hora da comparação
=end
for i in 0..19
  for j in 0..14
    num[i][j] = a
  end
end

=begin
  declarando variaveis booleana para sair do whilhe
=end
flag = false
varExiste = false

#while para controle para perguntar se que fazer uma nova agendação
while flag == false

  #perguntando sobre:
  #qual seu nome;
  #fila;
  #e coluna.
  puts "Qual o seu nome? "
  nome = gets.chomp.to_s

  puts "Fila: "
  fila = gets.chomp.to_i
```

```
puts "Coluna: "
coluna = gets.chomp.to_i

#if para validar se já não existe uma pessoa no lugar
if num[fila][coluna] != a
  varExiste = true
end

#while para validar os dados novamente caso ja exista uma pessoa no lugar
while varExiste == true

  #if para mostra a mensagem que já existe
  if varExiste == true
    puts "já tem um numero existente!"
    puts ""
  end

  #refeita a pergunta sobre:
  #qual seu nome;
  #fila;
  #e coluna.
  puts "Qual o seu nome? "
  nome = gets.chomp.to_s

  puts "Fila: "
  fila = gets.chomp.to_i

  puts "Coluna: "
  coluna = gets.chomp.to_i

  #if para validar se já não existe uma pessoa no lugar de novo
  if num[fila][coluna] != a
    varExiste = true
  else
```

```
    else
      varExiste = false
    end
  end

  #if para colocar a pessoa no lugar que ela desejou
  if nome != "" &&  num[fila][coluna] == a
    num[fila][coluna] = "poltrona do: #{nome}"
  end

  #zerandoa as variaveis
  nome = ""
  flag = false
  varExiste = false
  varMaiAg = false
  a = 1

  #while para perguntar se deseja fazer uma nova agendação
  while varMaiAg == false
    puts "deseja fazer mais alguma agendação: \ns => Sim\nn => Não"
    aux = gets.chomp.to_s
    #if para validar a resposta
    if aux == "s" || aux == "n"
      varMaiAg = true
    end
  end
  #if para sair do while de pergunta se quer uma nova agendação
  #Que está localizado na linha 24
  if aux == "s"
    flag= false
  elsif aux == "n"
    flag = true
  end
end
```

```
    end
  end
  #if para sair do while de pergunta se quer uma nova agendação
  #Que está localizado na linha 24
  if aux == "s"
    flag= false
  elsif aux == "n"
    flag = true
  end
end

#zerandoa as variaveis
a = 1
none = 1

#criando dois for para colocar numeração nas poltronas
for i in 0..19
  for j in 0..14
    if num[i][j] == none
      num[i][j] = a
    end
    a+=1
  end
end

#criando o for para printar no console que está em cada lugar
for i in 0..19
  for j in 0..14
    #o print é para printar na tela e fica na linha
    print "  #{num[i][j]}  ||"
  end
  #puts para pular linha
  puts""
end
```

3. CAPÍTULO: INTRODUZINDO INTERFACE GRÁFICA EM RUBY

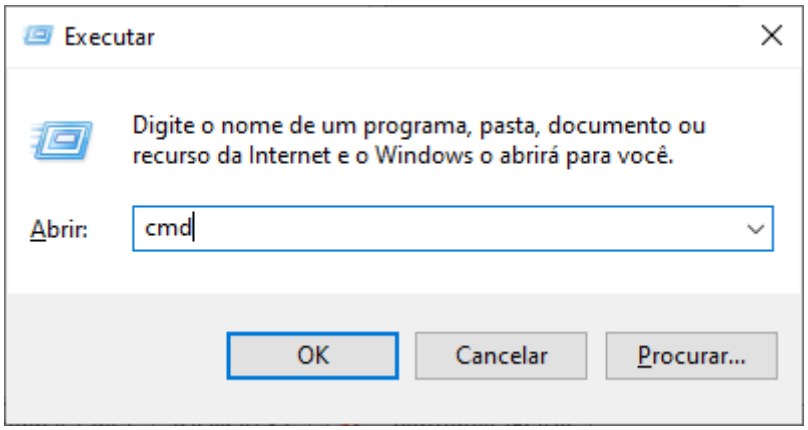
Interface Gráfica é um conceito da forma de interação entre o usuário do computador e um programa por meio de uma tela ou representação gráfica, visual, com desenhos, imagens, etc. Geralmente é entendido como a “tela” de um programa. Expressões como “buttom”, “labels”, “radiobuttom” fazem parte da dinâmica de uma interface gráfica, isso é, por meio dessa há a possibilidade de executar os comandos de forma lúdica e aprimorada ao usuário, diferindo das linhas de código.

Dessa forma, neste capítulo será apresentada uma das formas que utiliza interface gráfica em conjunto com a linguagem de programação Ruby, destacando sua utilização na IDE Atom, usando como base os programas exemplificados anteriormente.

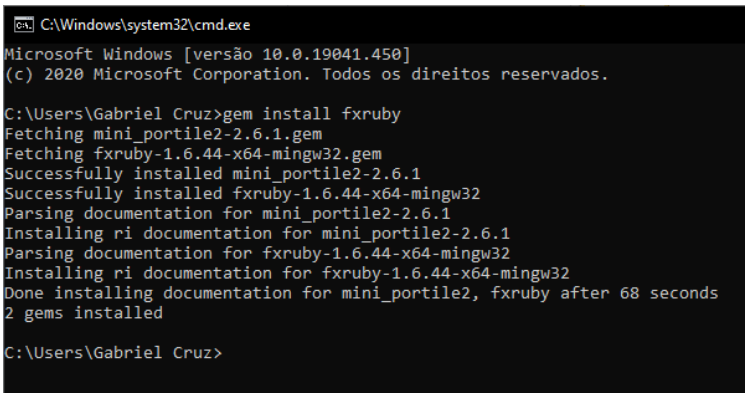
3.1. RECURSOS PARA INTERFACE GRÁFICA EM RUBY

Oposta a outras linguagens de programação, a instalação de uma GEM, é fundamental para a utilização de interface gráfica em Ruby, já que a GEM éum gerenciador de pacotes muito avançado e flexível do Ruby, tal como biblioteca de código reutilizável do Ruby**.** Assim, diferente da linguagem C#, por exemplo, que já possui uma estrutura gráfica definida, o Windows form, o Ruby necessita de uma GEM para execução gráfica, aqui será feito uso da chamada de “fxruby”. Abaixo segue o passo a passo da instalação para, posteriormente, fazer-se uso da interface gráfica.

1) Primeiramente, tecle win + r para abrir o menu “executar ” do Windows. Em seguida digite “cmd” no espaço apresentado e aperte “ok”.



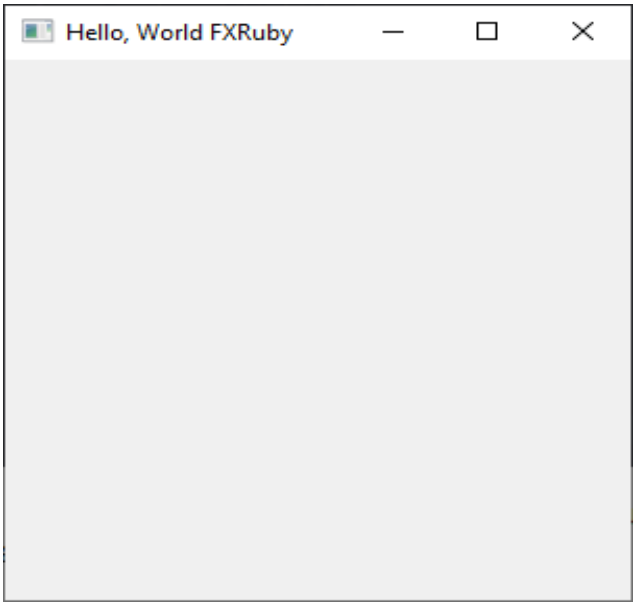
2) Agora, com o cmd aberto, utilize o comando: “geminstallfxruby” e tecle “enter”; esperar fazer a instalação dos arquivos necessários.



3) Feito isso, pode-se fechar o cmd. Agora, abra seu editor de textos e crie um novo projeto. No seu editor de textos com o projeto aberto, digite as seguintes linhas de comando:

```
1
2  require 'fox16'
3  include Fox
4  class Fxruby < FXMainWindow
5    def initialize(app)
6      super(app, "Hello, World FXRuby", :width => 300, :height => 300 )
7  end
8  def create
9    super
10   show(PLACEMENT_SCREEN)
11 end
12 end
13 app = FXApp.new
14 Fxruby.new(app)
15 app.create
16 app.run
17
```

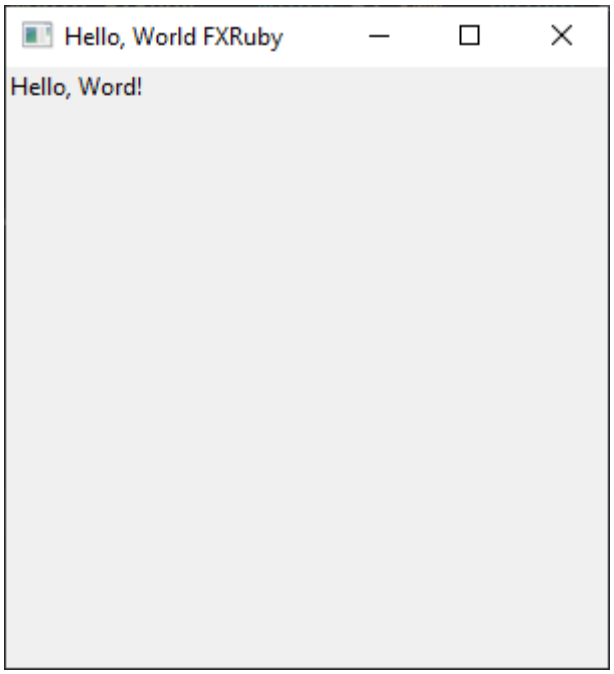
4) Com as linhas de comando acima, ao executarmos, aparecerá a seguinte janela:



5) E, para gerar o famoso “Hello, World”, basta acrescentar esta linha de comando “FXLabel.new(self, "Hello, Word!")” deste modo:

```
require 'fox16'
include Fox
class Fxruby < FXMainWindow
  def initialize(app)
    super(app, "Hello, World FXRuby", :width => 300, :height => 300 )
    FXLabel.new(self,  "Hello, Word!")
end
def create
  super
  show(PLACEMENT_SCREEN)
end
end
app = FXApp.new
Fxruby.new(app)
app.create
app.run
```

6) Ao executar:



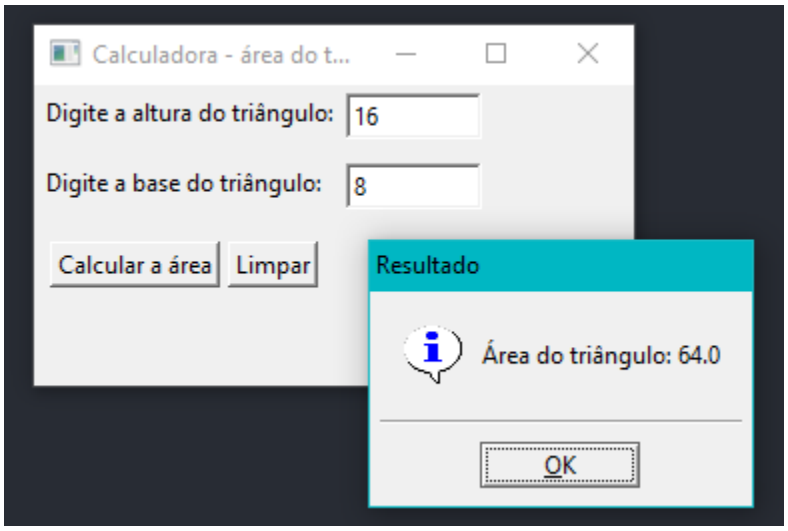
3.2. INTERFACE GRÁFICA NA PRÁTICA

3.2.1. ÁREA DE UM TRIÂNGULO – BUTTON, MENSSAGEBOX E TEXTFIELDS

Esse programa tem como finalidade calcular a área de um triângulo a partir da digitação de sua altura e base. Foram utilizados dois“buttons” (botões), um com a função de apresentar ao usuário o resultado do cálculo da área do triângulo e outro com a função de remover o que foi escrito nas duas “textfields” (campos de textos).

Ao acionar o “button”: “Calcular a área”, é apresentada uma “menssagebox” (caixa de mensagem) com o valor da área do triângulo, mas essa “messagebox” poderia ter sido substituída por uma “label” (rótulo de texto) dentro do form.

```ruby
/Biblioteca necessária para a formação do formulário /
require 'fox16'
include Fox
class Fxruby < FXMainWindow
  def initialize(app)

    /Formatação do form/
    super(app, "Calculadora - área do triângulo", :width => 300, :height => 150)

    /Objetos do formulário/
    hFrame1 = FXHorizontalFrame.new(self)
    label = FXLabel.new(hFrame1, "Digite a altura do triângulo:")
    @valorAltura = FXTextField.new(hFrame1, 10)

    hFrame2 = FXHorizontalFrame.new(self)
    label = FXLabel.new(hFrame2, "Digite a base do triângulo:  ")
    @valorBase = FXTextField.new(hFrame2, 10)

    vFrame1 = FXVerticalFrame.new(self, :opts => LAYOUT_FILL)
    hFrame3 = FXHorizontalFrame.new(vFrame1)
    botaoCalcular = FXButton.new(hFrame3, "Calcular a área")
    botaoLimpar = FXButton.new(hFrame3, "Limpar")

    /Ao acionar o botão limpar, todas as caixas de texto ficarão vazias para uma nova consulta/
     botaoLimpar.connect(SEL_COMMAND) do
       @valorAltura.text = ""
       @valorBase.text = ""
     end

    /Quando o usuário clicar no botão calcular, irá abrir uma nova janela com o resultado do cálculo
    da área do triângulo/
    botaoCalcular.connect(SEL_COMMAND) do
      resultado = ((@valorAltura.text.to_f * @valorBase.text.to_f) / 2).to_s
      FXMessageBox.information(app, MBOX_OK, "Resultado", "Área do triângulo: #{resultado}")
end

end
  def create
    super
    show(PLACEMENT_SCREEN)
  end
end
app = FXApp.new
/Chamando a classe FXRuby/
Fxruby.new(app)
/Criando o form/
app.create
/Mantendo o form até que ele seja encerrado/
app.run
```



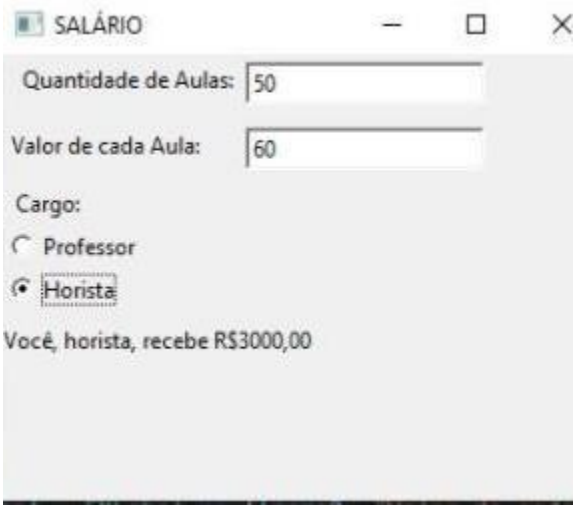
3.2.2. CÁLCULO DE SALÁRIO - RADIOBUTTON

Esse programa tem objetivo de calcular o salário de um horista ou de um professor. Nota-se que, além dos elementos utilizados no item anterior, o elemento gráfico escolhido foi o “RadioButton”, pois com ele o usuário pode escolher entre opções (opção horista ou professor).

```
=begin
Aqui estou chamndo a gem fox16
e solicitando a classe Fox
=end
require 'fox16'
include Fox
=begin
    aqui estou criando a classe Fxruby
=end
class Fxruby < FXMainWindow

    def initialize(app)
        super(app, "SALÁRIO", :width => 300, :height => 300)

        #crindo caixas "invisiveis" para uma melhor organização dos componentes
        frame = FXHorizontalFrame.new(self)
        frame2 = FXHorizontalFrame.new(self)

        #aqui estou criando o label
        quantidadeLabel = FXLabel.new(frame, "  Quantidade de Aulas:")
        valorLabel = FXLabel.new(frame2, "Valor de cada Aula:      ")

        #aqui estou criando o text para receber os valores
        quantidade = FXTextField.new(frame, 20)
        valor = FXTextField.new(frame2, 20)

        #aqui estou criando o grupo para colocar o radionbutton
        #juntamento com o data que vai dar um valor para cara radionbutton
        grupo = FXGroupBox.new(self, "Cargo: ")
        data = FXDataTarget.new(2)

        #aqui estou criando o radionbutton
        prof = FXRadioButton.new(grupo, "Professor", data, FXDataTarget::ID_OPTION)
        horista = FXRadioButton.new(grupo, "Horista", data, FXDataTarget::ID_OPTION + 1)
```

```ruby
        #aqui estou criando o grupo para colocar o radionbutton
        #juntamento com o data que vai dar um valor para cara radionbutton
        grupo = FXGroupBox.new(self, "Cargo: ")
        data = FXDataTarget.new(2)

        #aqui estou criando o radionbutton
        prof = FXRadioButton.new(grupo, "Professor", data, FXDataTarget::ID_OPTION)
        horista = FXRadioButton.new(grupo, "Horista", data, FXDataTarget::ID_OPTION + 1)

        #criando variaveis para fazer os calculos
        salario = 0
        salarioLabel = FXLabel.new(self , " ")

        #aqui chamando ação do butão caso o usário clique
        data.connect(SEL_COMMAND) do
            #if para testar a condição
            if data.value == 0
                #claculo para o professor
                salario = quantidade.text.to_i * valor.text.to_i * 1.25
                #aqui vai "imprimir" no label o valor do salário do professor
                salarioLabel.text = "Você, professor, recebe R$#{salario.to_s}0"
            else
                #calculo para o horista
                salario = quantidade.text.to_i * valor.text.to_i
                #aqui o salário do horista vai ser colocado no label
                salarioLabel.text = "Você, horista, recebe R$#{salario.to_s},00"
            end
        end
    end

    def create
            super
            show (PLACEMENT_SCREEN)
    end
```

```ruby
        #aqui chamando ação do butão caso o usário clique
        data.connect(SEL_COMMAND) do
            #if para testar a condição
            if data.value == 0
                #claculo para o professor
                salario = quantidade.text.to_i * valor.text.to_i * 1.25
                #aqui vai "imprimir" no label o valor do salário do professor
                salarioLabel.text = "Você, professor, recebe R$#{salario.to_s}0"
            else
                #calculo para o horista
                salario = quantidade.text.to_i * valor.text.to_i
                #aqui o salário do horista vai ser colocado no label
                salarioLabel.text = "Você, horista, recebe R$#{salario.to_s},00"
            end
        end
    end

    def create
            super
            show (PLACEMENT_SCREEN)
    end
end

#chamando o forme
app = FXApp.new
#chamando a classe FXruby
Fxruby.new(app)
#criando o forme
app.create
#mantendo o forme
app.run
```



3.2.3. CÁLCULO DE SALÁRIO - COMBOBOX

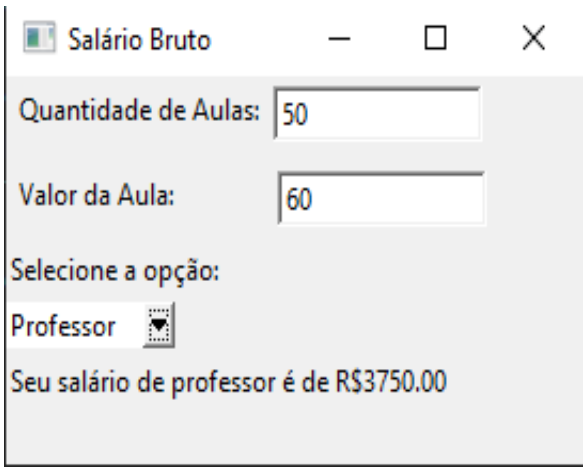
Agora você, aprendiz leitor, já sabe como criar um formulário que calcule o salário por meio de “RadioButton”, entretanto é possível repetir o mesmocódigo, mas utilizando o elemento gráfico: “combobox”.

Desse modo, foi utilizado o "combobox” no programa para mostrar outra forma de se fazer lista em forms. Vale ressaltar que a maior praticidade do "combobox”é que não tem limite, por isso, é usado para listas muito extensas, diferente do “radiobutton” que é mais prático para listas de "sim ou não", simples e práticas.

```
=begin
Aqui estou chamndo a gem fox16
e solicitando a classe Fox
=end
require 'fox16'
include Fox
=begin
    aqui estou criando a classe Fxruby
=end
class Fxruby < FXMainWindow

  def initialize(app)
    super(app, "Salário Bruto", :width => 270, :height => 160)

    #crindo caixas "invisiveis" para uma melhor organização dos componentes
    hFrame1 = FXHorizontalFrame.new(self)
    frame2 = FXHorizontalFrame.new(self)

    #estou criando o label e o text dentro no primeiro frame (hFrame1)
    #esse é o text e o label da quantidade
    qtdAula = FXLabel.new(hFrame1, "Quantidade de Aulas:")
    qtd = FXTextField.new(hFrame1, 15)
    #já aqui estou criando o label e o text no segundo frame (frame2)
    #já esse é o text e o label do valor
    vlrAula = FXLabel.new(frame2 , "Valor da Aula:              " )
    vlr = FXTextField.new(frame2, 15)

    #criando a variavel
    sb = 0
    #um label para a escola do cargo
    label = FXLabel.new(self, "Selecione a opção:")

    #width limite para escolha
    width = 10
```

```
    #width limite para escolha
    width = 10
    #criando o combobox
    cbo = FXComboBox.new(self, width, nil, 2, COMBOBOX_STATIC)
    #colocando os intens no combobox
    cbo.appendItem(" ",0)
    cbo.appendItem("Horista",1)
    cbo.appendItem("Professor", 2)

    #o tanto de intens que iram ficar visivel
    cbo.numVisible = 3
    salario = FXLabel.new(self, "")

    #aqui chamando ação do butão caso o usário clique
    cbo.connect(SEL_COMMAND) do
      #if para testar a condição
      if cbo.text == "Horista"
        #claculo para o horista
        sb = vlr.text.to_i * qtd.text.to_i
        #aqui vai "imprimir" no label o valor do salário do professor
        salario.text = "Seu salário de horista é de R$#{sb},00"
      #elsif para testar se foi escolhido o profesor
      elsif cbo.text == "Professor"
        #claculo para o professor
        sb = vlr.text.to_i * qtd.text.to_i * 1.25
        #aqui vai "imprimir" no label o valor do salário do professor
        salario.text = "Seu salário de professor é de R$#{sb}0"
      end
    end
  end
  def create
    super
    show (PLACEMENT_SCREEN)
  end
```

```
    salario = FXLabel.new(self, "")

    #aqui chamando ação do butão caso o usário clique
    cbo.connect(SEL_COMMAND) do
      #if para testar a condição
      if cbo.text == "Horista"
        #claculo para o horista
        sb = vlr.text.to_i * qtd.text.to_i
        #aqui vai "imprimir" no label o valor do salário do professor
        salario.text = "Seu salário de horista é de R$#{sb},00"
      #elsif para testar se foi escolhido o profesor
      elsif cbo.text == "Professor"
        #claculo para o professor
        sb = vlr.text.to_i * qtd.text.to_i * 1.25
        #aqui vai "imprimir" no label o valor do salário do professor
        salario.text = "Seu salário de professor é de R$#{sb}0"
      end
    end
  end
  def create
    super
    show (PLACEMENT_SCREEN)
  end
end

#chamando o forme
app = FXApp.new
#chamando a classe FXruby
Fxruby.new(app)
#criando o forme
app.create
#mantendo o forme
app.run
```

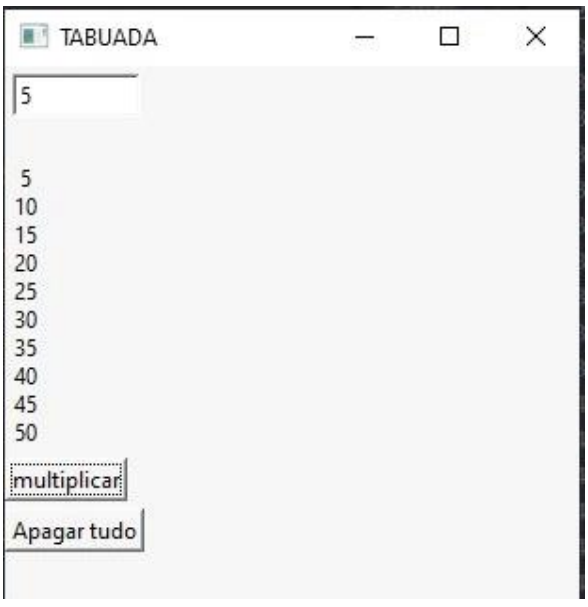


3.2.4. CÁLCULO DE TABUADA

No seguinte programa, tal como nos exemplos anteriores, foi colocado um botão através da função “button” com o intuito de realizar procedimentos para efetuar uma tabuada. Ao utilizar esse “button”, o usuário terá que colocar o número que deseja ver a tabuada em uma caixa de texto,”TextField”. Após isso, em texto, aparecerão os valores da tabuada do número requerido. Em sequência, outro “button” é colocado, esse com a finalidade de apagar dados de registros feitos anteriormente dentro do programa.

```ruby
#Chamando a gem fox16
require 'fox16'
include Fox

#Criação da classe FXRuby
class Fxruby < FXMainWindow
    def initialize(app)
        super(app, "TABUADA", :width => 200, :height => 200)

#criação de uma Frame para organização
        hFrame1 = FXHorizontalFrame.new(self)

#Criação de um textfield para que o usuário digite o valor desejado
        num = FXTextField.new(hFrame1, 10)
#Criação de uma label que mostrará a tabuada do valor
        label = FXLabel.new(self, "")
#Criação de uma variável auxiliar
        aux = 1
# Criação de uma string que receberá os valores da tabuada
        tabuada = " "

#Criação de um botão MULTIPLICAR
        button = FXButton.new(self,'multiplicar')

#Ação do botão Multiplicar que irá criar o laço de multiplicação e
# inserir os resultados na label
        button.connect(SEL_COMMAND) do
                while aux <= 10
                        valor = (num.text.to_i) * aux
                        tabuada = tabuada.to_s + "\n #{num.text} X #{aux} = #{valor.to_s}"
                        aux = aux + 1
                end
                label.text = tabuada
        end
```

```ruby
# Criação do botão APAGAR TUDO
        button3 = FXButton.new(self, 'Apagar tudo')
#Ação do Botão APAGAR TUDO que tornará tudo novamente vazio
        button3.connect(SEL_COMMAND) do
                label.text = ''
                num.text = ''
                aux = 1
                tabuada = " "
        end
end

def create
        super
        show (PLACEMENT_SCREEN)
end
end

#Chamando o Form
app = FXApp.new
#Chamando a Classe
Fxruby.new(app)
#Criando o Form
app.create
#Mantendo o Form aberto
app.run
```



4. CAPÍTULO: PROGRAMAÇÃO ORIENTADA A OBJETO (POO)

Já sabemos como criar variáveis, laços de repetição, laços de decisão, forms, mas ainda temos muito a explorar na programação. É perceptível que todo programa sempre terá alguma forma de verificar se determinado dado inserido ou não é válido, por exemplo, se o usuário digitou números da área destinada ao celular,se o cliente selecionou uma das opções disponíveis necessárias para a próxima etapa e assim por diante. Ademais, muitas vezes a verificação de dados é necessária em mais de uma parte do programa, fazendo com que ele fique mais extenso do que o necessário - dando margem para erros e dificuldade na manutenção.

Pensando nisso, há uma forma de "chamar" tal código de análise sem precisar repeti-lo a partir de classe e de objeto.

A classe é uma maneira de definição de linguagem orientada a objeto, a qual pode ser definida por atributos e seu comportamento é definido por métodos. Abaixo há um exemplo de criação de classe, na qual necessita de 3 argumentos de entrada e exibe uma resposta ao final da classe. Note, portanto, que pode-se criar diferentes tipos de classe, com diferentes finalidades, isso é, classes são feitas não exclusivamente para tratar de possíveis erros de validação.

```
#Classe para herdar as informações digitadas pelo usuário
class CalcularArea < FXMainWindow
  def initialize(app, n1, n2)
     resultado = ((n1.text.to_f * n2.text.to_f) / 2).to_s
     FXMessageBox.information(app, MBOX_OK,"RESULTADO", "Área do triângulo: #{resultado}")
  end
```

Acima está a criação da classe, porém como "chamá-la"/ inicializá-la? Para isso, temos o objeto. O objeto é uma entidade da classe, ou seja, quando queremos utilizar a classe, criamos um novo objeto dela, como demonstrado a seguir.

```
#Criação do objeto de instância para receber os valores e utilizá-los na classe AreaTriangulo
obj = CalcularArea.new(app, @valorAltura, @valorBase)
```

Além disso, é importante dizer que se o programador utiliza muito uma classe em vários programas, ele pode implementá-la na biblioteca. Logo, não precisará escrever o escopo da classe nesses variados projetos, só necessitará criar o objeto dela.

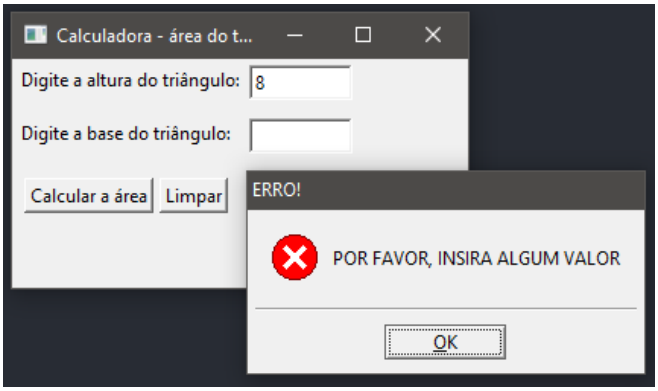
Por fim, esse capítulo terá a função de transformar os programas já criados em orientados a objetos.

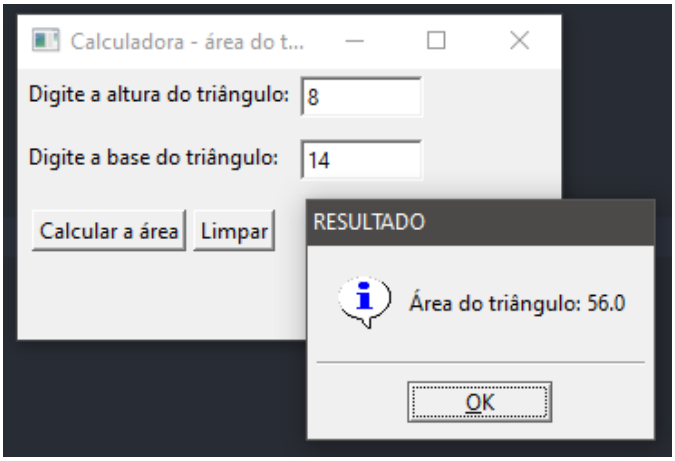
4.1. ÁREA DO TRIÂNGULO – (POO)

```ruby
#Biblioteca para a formação do formulário
require 'fox16'
include Fox

#Classe erro
class Erro < FXMainWindow
    def initialize(app)
        FXMessageBox.information(app, MBOX_OK,"ERRO!", "CAIXA DE TEXTO VAZIA... POR FAVOR, INSIRA ALGUM VALOR")
    end
  end

#Classe para herdar as informações digitadas pelo usuário
class CalcularArea < FXMainWindow
  def initialize(app, n1, n2)
     resultado = ((n1.text.to_f * n2.text.to_f) / 2).to_s
     FXMessageBox.information(app, MBOX_OK,"RESULTADO", "Área do triângulo: #{resultado}")
  end
end

#Classe para limpar os campos
class Limpar < FXMainWindow
  def initialize(app, n1, n2)
    n1.text = ""
    n2.text = ""
  end
end
```

```ruby
#Classe principal
class FXRuby < FXMainWindow
  def initialize(app)
    super(app, "Calculadora - área do triângulo", :width => 300, :height => 150)

    /Objetos do formulário/
    hFrame1 = FXHorizontalFrame.new(self)
    label = FXLabel.new(hFrame1, "Digite a altura do triângulo:")
    @valorAltura = FXTextField.new(hFrame1, 10)

    hFrame2 = FXHorizontalFrame.new(self)
    label = FXLabel.new(hFrame2, "Digite a base do triângulo:  ")
    @valorBase = FXTextField.new(hFrame2, 10)

    vFrame1 = FXVerticalFrame.new(self, :opts => LAYOUT_FILL)
    hFrame3 = FXHorizontalFrame.new(vFrame1)
    botaoCalcular = FXButton.new(hFrame3, "Calcular a área")
    botaoLimpar = FXButton.new(hFrame3, "Limpar")


    botaoCalcular.connect(SEL_COMMAND) do
      #Condição para verificar se os campos estão vazios para chamar a classe erro
      if @valorAltura.text == ""
          obj = Erro.new(app)
        else
          if @valorBase.text == ""
            obj = Erro.new(app)
        else
```

```ruby
          #Criação do objeto de instância para receber os valores e utilizá-los na classe Arealriangulo
          obj = CalcularArea.new(app, @valorAltura, @valorBase)
    end
  end
end

    #Objeto de instância para utilizar na classe Limpar
    botaoLimpar.connect(SEL_COMMAND) do
      obj = Limpar.new(app, @valorAltura, @valorBase)
    end
end

def create
  super
    show (PLACEMENT_SCREEN)
  end
end
#Chamando o Form
app = FXApp.new
#Chamando a Classe
FXRuby.new(app)
#Criando o Form
app.create
#Mantendo o Form aberto
app.run
```





Esse programa, como já mencionado tem como finalidade calcular a área de um triângulo a partir da digitação de sua altura e base, ondeforam utilizados “buttons” (botões), “textfields” e uma “menssagebox”.

Nessa nova versão do programa, com o intuito de mostrar como se implementa a arquitetura POO, foramutilizadas três classes:

1- Classe Erro: onde se é chamada apenas quando o programa detecta que não foi digitado nenhum valor em uma ou nas duas textfields, apresentando assim uma menssagebox informando o erro;

2- Classe CalcularArea: recebe (herda) os valores digitados pelo usuário;

3- Classe Limpar:utilizada para limpar as textfields quando o button limpar for requisitado.

Além dessas três classes, também houve a implementaçãoda classe principal que apresenta os objetos do formulário. Para cada uma dessas classes foi preciso criar um objeto para instancia-las, no caso da classe erro foi necessário colocar o objeto em uma condição if/else para verificar se os campos textfields estavam vazios.

4.2. PROFESSOR/HORISTA – (POO)

```ruby
#Chamando a gem Fox
require 'fox16'
include Fox

class Resultado < FXMainWindow
    def initialize(app, tipo, sal)
        if tipo == "horista"
            FXMessageBox.information(app, MBOX_OK,"SALÁRIO", "Seu salário de #{tipo} é de R$#{sal},00")
        else
            FXMessageBox.information(app, MBOX_OK,"SALÁRIO", "Seu salário de #{tipo} é de R$#{sal}0")
        end
    end
end

#Classe Erro como caixa de mensagem
class Erro < FXMainWindow
    def initialize(app)
        FXMessageBox.information(app, MBOX_OK,"ERRO!", "CAIXA DE TEXTO VAZIA... POR FAVOR, INSIRA ALGUM VALOR")
    end
end
```

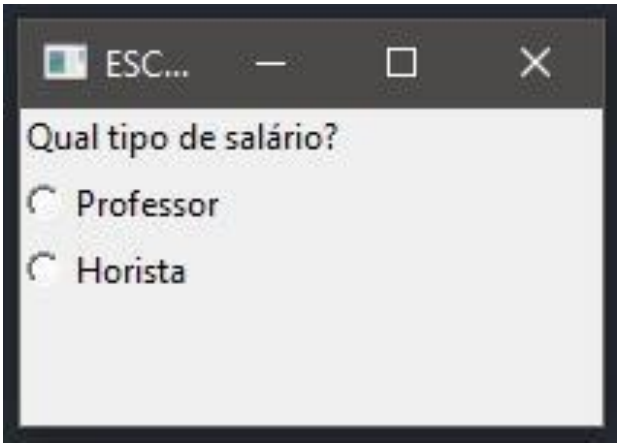
```ruby
#Classe Horista
class Salario_Horista < FXMainWindow
    def initialize(app)
        super(app, "HORISTA", :width => 400, :height => 400)
        #Criando uma variável para receber a função app
        chamar = app
        frame = FXHorizontalFrame.new(self)
        frame2 = FXHorizontalFrame.new(self)

        quantidadeLabel = FXLabel.new(frame, "  Quantidade de Aulas:")
        valorLabel = FXLabel.new(frame2, "Valor de cada Aula:      ")

        #Criando as caixas de texto que receberão os valores
        quantidade = FXTextField.new(frame, 20)
        valor = FXTextField.new(frame2, 20)

        salario = 0
        salarioLabel = FXLabel.new(self , "")

        button = FXButton.new(self, "CALCULAR")

        #Ação do Botão Calcular
        button.connect(SEL_COMMAND) do
            if quantidade.text == ""
                #Chamando a Classe Erro
                h = Erro.new(chamar)
            else
                if valor.text == ""
                    #Chamando a Classe Erro
                    h = Erro.new(chamar)
                else
                    salario = quantidade.text.to_i * valor.text.to_i
                    r = Resultado.new(chamar, "Horista", salario)
                end
```

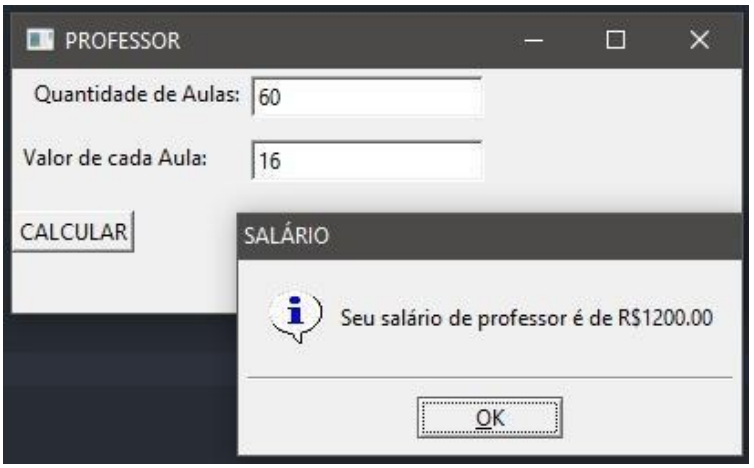
```
            end

        end
    end

    def create
            super
            show (PLACEMENT_SCREEN)
    end
end

#Classe Professor
class Salario_Professor < FXMainWindow
    def initialize(app)
        super(app, "PROFESSOR", :width => 400, :height => 400)
        #Criando uma variável para receber a função app
        chamar = app

        frame = FXHorizontalFrame.new(self)
        frame2 = FXHorizontalFrame.new(self)

        #Criando as caixas de texto que receberão os valores
        quantidadeLabel = FXLabel.new(frame, "  Quantidade de Aulas:")
        valorLabel = FXLabel.new(frame2, "Valor de cada Aula:      ")

        quantidade = FXTextField.new(frame, 20)
        valor = FXTextField.new(frame2, 20)

        salario = 0
        salarioLabel = FXLabel.new(self , "")

        button = FXButton.new(self, "CALCULAR")
```

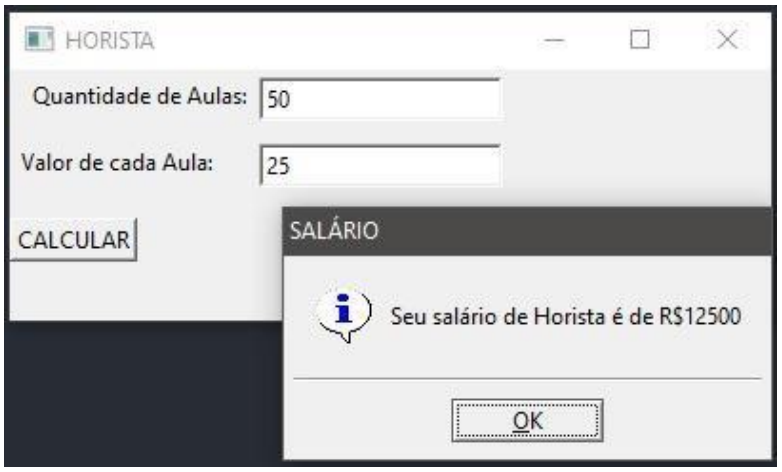
```
        #Ação do Botão Calcular
        button.connect(SEL_COMMAND) do
            if quantidade.text == ""
                #Chamando a Classe Erro
                h = Erro.new(chamar)
            else
                if valor.text == ""
                    #Chamando a Classe Erro
                    h = Erro.new(chamar)
                else
                    salario = quantidade.text.to_i * valor.text.to_i * 1.25
                    r = Resultado.new(chamar, "professor", salario)
                end

            end

        end
    end


    def create
            super
            show (PLACEMENT_SCREEN)
    end
end
```

```ruby
#Classe Inicial
class Inicio < FXMainWindow
        def initialize(app)
            super(app, "ESCOLHA", :width => 200, :height => 200)
            #Criando uma variável para receber a função app
            chamar = app

            label = FXLabel.new(self, "Qual tipo de salário?")

            #Criando os Rádiobuttons
            data = FXDataTarget.new(2)
            prof = FXRadioButton.new(self, "Professor", data, FXDataTarget::ID_OPTION)
            horista = FXRadioButton.new(self, "Horista", data, FXDataTarget::ID_OPTION + 1)

            #Ação do Data
            data.connect(SEL_COMMAND) do
                if data.value == 0
                    #Abrindo a Classe Professor
                    h = Salario_Professor.new(chamar)
                    h.create
                    h.show(PLACEMENT_SCREEN)
                    #Fechando a Classe atual
                    self.destroy
                else
                    #Abrindo a Classe Professor
                    h = Salario_Horista.new(chamar)
                    h.create
                    h.show(PLACEMENT_SCREEN)
                    #Fechando a Classe atual
                    self.destroy
                end
            end
        end
```

```ruby
def create
    super
    show (PLACEMENT_SCREEN)
end
end


#Chamando o Form
app = FXApp.new
#Chamando a Classe
Inicio.new(app)
#Criando o Form
app.create
#Mantendo o Form aberto
app.run
```







Como já foi indicado, o programa acima tem a função e objetivo de calcular o salário de um professor ou de um horista. Por meio de TextFields e variáveis há o armazenamento dos dados inseridos pelo usuário. Como visto ao decorrer deste artigo, utilizaram-se laços de repetições, IF/ELSE, para validar e calcular o resultado correto, o qual é exibido em uma caixa de texto.

Nesse caso, as classes são utilizadas para agilizar o processo de execução do programa, de forma que:

1- **Classe erro:** ao ser solicitada, exibe em uma caixa de mensagem o erro “CAIXA DE TEXTO VAZIA”, indicando ao usuário qual etapa é estritamente necessária para o processo e evitando erros no desenvolvimento dos próximos passos do código.
2- **Classe inicio:** utilizada para instanciar o “form” ao ser requisitada pelo objeto. Ou seja, sempre que for necessário o uso do “form” de escolha, basta fazer uso da classe inicio. Nessa classe estão presentes laços de repetição que permitem a exibição da classe Salario_Horista e Salario_Professor.

3- **Classe Salario_Horista:** essa é requisitada por meio da classe inicio, nela há o calculo correto para o salário de um horista.
4- **Classe Salario_Professor:** essa é requisitada por meio da classe inicio, nela há o calculo correto para o salário de um horista.
5- **Classe Resultado:** por meio dos processos feitos nas classes Salario_Professor e Salario_Horista, exibe-se os valores obtidos em uma “MessageBox”.

Interessante citar que as classes apresentam parâmetros para sua inicialização. “App” é um exemplo de parâmetro de entrada de dado, ele é utilizado em todas as classes e tem um valor fixo, armazenando um dos valores digitados pelo usuário, crucial para o desenvolvimento da aplicação.

Nota-se que, para chamar uma Classe, é importante que haja o instanciamento deum objeto para a mesma, o qual redirecionará seus valores e permitirá seu uso, tal como no exemplo:

```
#Abrindo a Classe Professor
h = Salario_Horista.new(chamar)
h.create
h.show(PLACEMENT_SCREEN)
#Fechando a Classe atual
self.destroy
```

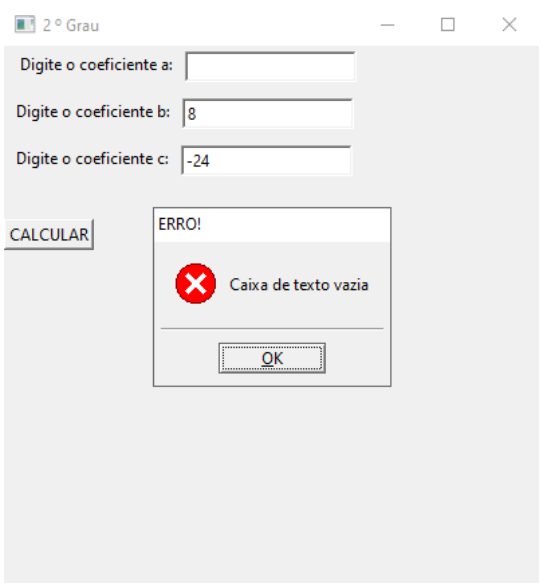
Onde h é o objeto para a abertura da classe Salario_Horista(chamar), que tem“chamar” como parâmetro de entrada de dado.

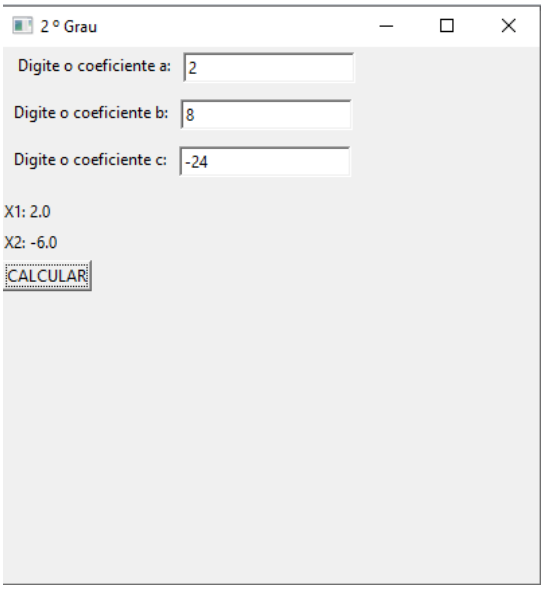
4.3. EQUAÇÃO DE SEGUNDO GRAU – (POO)

```ruby
require 'fox16'
include Fox

class Erro < FXMainWindow
  def initialize(app)
    FXMessageBox.error(app, MBOX_OK,"ERRO!", "Caixa de texto vazia")
  end
end

class Segundo < FXMainWindow
    def initialize(app)
        super(app, "2 º Grau", :width => 400, :height => 400)

        frame = FXHorizontalFrame.new(self)
        frame2 = FXHorizontalFrame.new(self)
        frame3 = FXHorizontalFrame.new(self)

        a = FXLabel.new(frame, "  Digite o coeficiente a: ")
        b = FXLabel.new(frame2, " Digite o coeficiente b: ")
        b = FXLabel.new(frame3, " Digite o coeficiente c: ")

        aa = FXTextField.new(frame, 20)
        bb = FXTextField.new(frame2, 20)
        cc = FXTextField.new(frame3, 20)

        delta = 0
        x1 = 0
        x2 = 0
        dn = FXLabel.new(self , "")
        raiz1 = FXLabel.new(self , "")
```

```ruby
        raiz2 = FXLabel.new(self , "")
        button = FXButton.new(self, "CALCULAR")

        button.connect(SEL_COMMAND) do
          if aa || bb || cc == ""
            obj = Erro.new(app)
          else

          end
          delta = (bb.text.to_i * bb.text.to_i) - ( 4 * aa.text.to_i * cc.text.to_i)

          if delta < 0
            dn.text = "Não existem raizes reais"
          else
            x1 = ((-bb.text.to_i + Math.sqrt(delta) )/(2 *aa.text.to_i))
            x2 = ((-bb.text.to_i - Math.sqrt(delta) )/(2 *aa.text.to_i))

            raiz1.text = "X1: #{x1}"
            raiz2.text = "X2: #{x2}"
          end

        end
    end

    def create
            super
            show (PLACEMENT_SCREEN)
    end
end
```

```
66
67  #Chamando o Form
68  app = FXApp.new
69  #Chamando a Classe
70  Segundo.new(app)
71  #Criando o Form
72  app.create
73  #Mantendo o Form aberto
74  app.run
75
```





O programa acima tem como objetivo executar uma equação de segundo grau composta por três coeficientes (a, b e c). Em sua execução, seria solicitado ao usuário que informasse o valor dos coeficientes correspondentes à equação. Após a recepção desses dados, o programa entraria em funcionamento exibindo o resultado dos cálculos sugeridos em uma caixa.

Em primeiro momento, a classe erro é posicionada para eventuais erros dentro do que será executado pelo usuário. Após isso, os coeficientes são declarados e a forma como a equação vai se desenvolver começa a ser moldada.

Colocou-se uma mensagem solicitando os valores de cada um dos coeficientes, após isso aparecem as variáveis presentes para a realização da equação.

Na segunda imagem de código, nota-se o botão para efetuar o cálculo, seguido das funções IF e ELSE. Os procedimentos de erro são colocados e a opção dos resultados também.

Têm-se presentes as seguintes funções:

1- **Math.sqrt:** para que a função aceite um valor e retorne a raiz quadrada do mesmo.
2- **IF e ELSE:** para que existam duas possibilidades ou mais de saída para aquela função.
3- **Classe Erro:** para que seja chamada quando não houver valor informado dentro dos coeficientes, fazendo com que apareça uma menssagebox informando o erro.

4.4. CADASTRO DE LIVROS – (POO)

Livro.rb

```ruby
require 'fox16'
include Fox
require_relative 'Codigo'

$ProcuarClass = Executar.new
$SalvarCLass = Executar.new
$LimparCLass = Executar.new
=begin
    Class para o visual
    classe para o cadastro
    class para a exclusão
    class para a pesquisa
=end
#chamando o forme
class Erro < FXMainWindow
  def initialize(app)
    FXMessageBox.error(app, MBOX_OK,"ERRO!", "Tem um epaço em branco!! :(")
  end
end

class Fxruby < FXMainWindow

    def initialize(app)
      super(app, "Cadastro Livro", :width => 400, :height => 300 )
      chamar = app
      #crindo caixas "invisiveis" para uma melhor organização dos componentes
      hFrame1 = FXHorizontalFrame.new(self)
      hframe2 = FXHorizontalFrame.new(self)
      hFrame3 = FXHorizontalFrame.new(self)
      hFrame4 = FXHorizontalFrame.new(self)
      hFrame5 = FXHorizontalFrame.new(self)
      hFrame6 = FXHorizontalFrame.new(self)
      #esse é o text e o label do codigo
      codigoLabel = FXLabel.new(hFrame1, "código:")
```

Livro.rb

```ruby
        hFrame4 = FXHorizontalFrame.new(self)
        hFrame5 = FXHorizontalFrame.new(self)
        hFrame6 = FXHorizontalFrame.new(self)
        #esse é o text e o label do codigo
        codigoLabel = FXLabel.new(hFrame1, "código:")
        codigo = FXTextField.new(hFrame1, 8)
        #já esse é o text e o label do titulo
        tituloLabel = FXLabel.new(hframe2 , "Titulo:   ")
        titulo = FXTextField.new(hframe2, 40)
        #esse é o text e o label do autor
        autorLabel = FXLabel.new(hFrame3, "Autor:   ")
        autor = FXTextField.new(hFrame3, 35)
        #já esse é o text e o label da editora
        editoraLabel = FXLabel.new(hFrame4 , "Editora: " )
        editora = FXTextField.new(hFrame4, 25)
        #já esse é o text e o label do ano
        anoLabel = FXLabel.new(hFrame5 , "Ano:      " )
        ano = FXTextField.new(hFrame5, 10)

        #os botões
        btnSalvar = FXButton.new(hFrame6, "Salvar" )
        btnPreocurar = FXButton.new(hFrame6, "Procurar" )
        btnLimpar = FXButton.new(hFrame6, "Limpar" )

        auxProcura = 1
        btnPreocurar.connect(SEL_COMMAND) do
          $ProcuarClass.Procurar(codigo, titulo, autor, editora, ano, auxProcura)
          auxProcura += 1
        end

        btnLimpar.connect(SEL_COMMAND) do
          $LimparCLass.Limpar(codigo, titulo, autor, editora, ano)
        end
```

Livro.rb

```ruby
        auxSalvar = 1
        btnSalvar.connect(SEL_COMMAND) do
          #Criando um if para viricar se está sem ser preencido
          if codigo == ""
            codigo = Erro.new(chamar)
            if titulo == ""
              titulo = Erro.new(chamar)
              if autor == ""
                autor = Erro.new(chamar)
                if editora == ""
                  editora = Erro.new(chamar)
                  if ano == ""
                    ano = Erro.new(chamar)
                  else
                      $SalvarCLass.Salvar(codigo, titulo, autor, editora, ano, auxSalvar)
                      auxSalvar += 1
                  end
                end
              end
            end
          end
        end
        def create
          super

          show (PLACEMENT_SCREEN)
        end
    end
```

Livro.rb

```
86          end
87        end
88        def create
89          super
90
91          show (PLACEMENT_SCREEN)
92        end
93    end
94    app = FXApp.new
95    #chamando a classe FXruby
96    Fxruby.new(app)
97    #criando o forme
98    app.create
99    #mantendo o forme
100   app.run
```

Codigo.rb

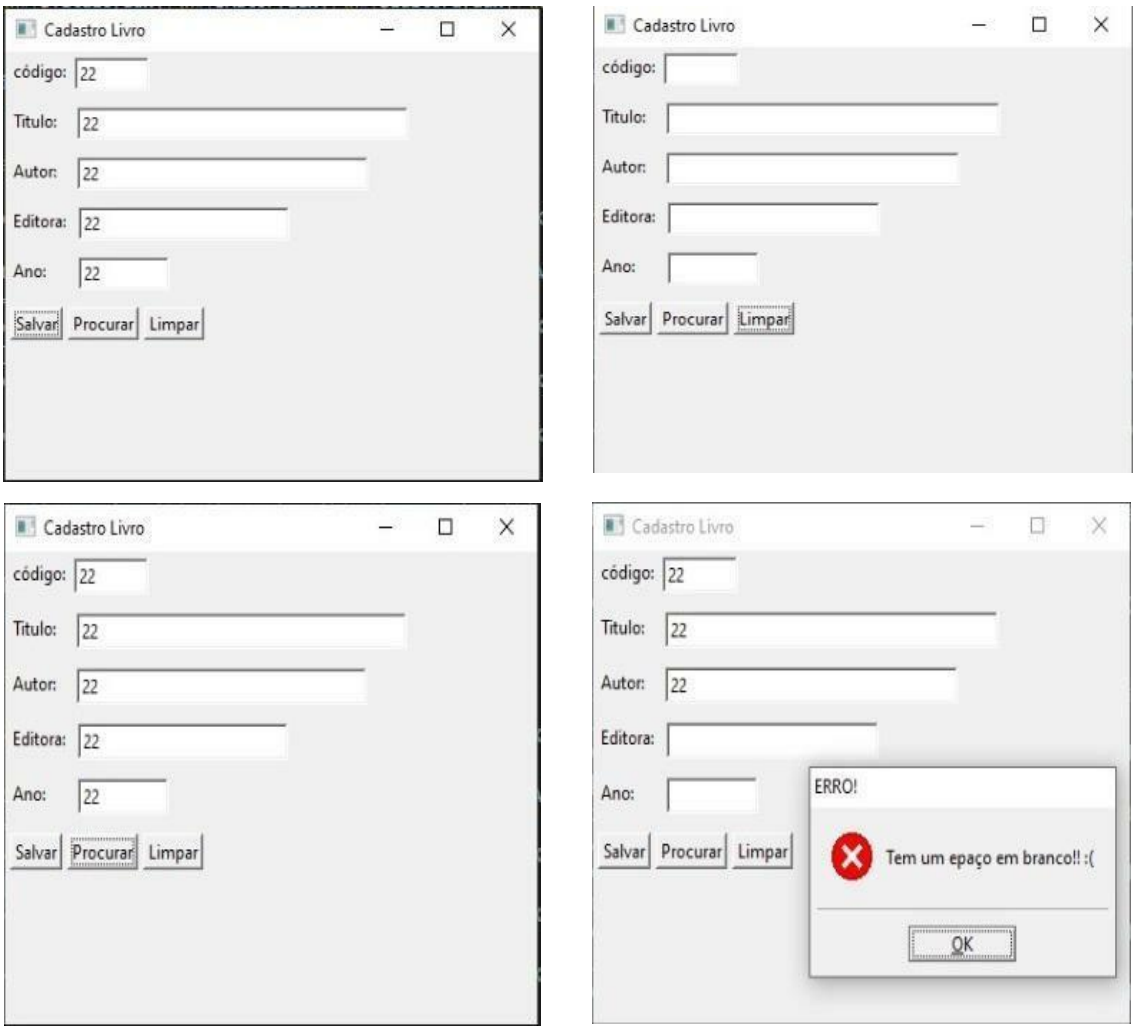
```
1   require 'fox16'
2   include Fox
3
4   class Executar
5
6     #colocando os vetores em global
7     $Codigo = Array.new(10)
8     $Titulo = Array.new(10)
9     $Autor = Array.new(10)
10    $Editora = Array.new(10)
11    $Ano = Array.new(10)
12
13    def Salvar(codLivro, tituLivro, autorLivro, editoLivro, anoLivro, aux)
14      #armazenando os valores
15      $Codigo[aux] = codLivro
16      $Titulo[aux] = tituLivro
17      $Autor[aux] = autorLivro
18      $Editora[aux] = editoLivro
19      $Ano[aux] = anoLivro
20      puts("#{$Codigo[aux]}\n#{$Titulo[aux]}\n#{$Autor[aux]}\n#{$Editora[aux]}\n#{$Ano[aux]}")
21    end
22
```

Codigo.rb

```ruby
    $Editora[aux] = editoLivro
    $Ano[aux] = anoLivro
    puts("#{$Codigo[aux]}\n#{$Titulo[aux]}\n#{$Autor[aux]}\n#{$Editora[aux]}\n#{$Ano[aux]}")
  end

  def Procurar(codTXTP, tituTXTP, autorTXTP, editoraTXTP, anoTXTP, vAUX)
    while vAUX < 10

      #verificando cada variavel caso a pessoa tenha colocado para buscar sem ser pelo o codigo
      if $Codigo[vAUX] == codTXTP
        return codTXTP.text = $Codigo[vAUX].to_s, tituTXTP.text = $Titulo[vAUX].to_s,autorTXTP.text= $Autor[vAUX].to_s, e
      elsif $Titulo[vAUX] == codTXTP
          return codTXTP.text = $Codigo[vAUX].to_s, tituTXTP.text = $Titulo[vAUX].to_s,autorTXTP.text= $Autor[vAUX].to_s,
      elsif $Autor[vAUX] == codTXTP
        return codTXTP.text = $Codigo[vAUX].to_s, tituTXTP.text = $Titulo[vAUX].to_s,autorTXTP.text= $Autor[vAUX].to_s, e
      elsif $Editora[vAUX] == codTXTP
        return codTXTP.text = $Codigo[vAUX].to_s, tituTXTP.text = $Titulo[vAUX].to_s,autorTXTP.text= $Autor[vAUX].to_s, e
      elsif $Ano[vAUX] == codTXTP
        return codTXTP.text = $Codigo[vAUX].to_s, tituTXTP.text = $Titulo[vAUX].to_s,autorTXTP.text= $Autor[vAUX].to_s, e
      end
    end
  end

  def Limpar(codTXT, tituTXT, autorTXT, editoraTXT, anoTXT)
    #limpando os textbox
    return codTXT.text="", tituTXT.text ="", autorTXT.text="", editoraTXT.text="", anoTXT.text =""
  end
end
```

Codigo.rb

```ruby

tor[aux]}\n#{$Editora[aux]}\n#{$Ano[aux]}")


oraTXTP, anoTXTP, vAUX)


nha colocado para buscar sem ser pelo o codigo
 tituTXTP.text = $Titulo[vAUX].to_s,autorTXTP.text= $Autor[vAUX].to_s, editoraTXTP.text= $Editora[vAUX].to_s, anoTXTP.text = $Ano[vAUX].to_s

s, tituTXTP.text = $Titulo[vAUX].to_s,autorTXTP.text= $Autor[vAUX].to_s, editoraTXTP.text= $Editora[vAUX].to_s, anoTXTP.text = $Ano[vAUX].to_s

 tituTXTP.text = $Titulo[vAUX].to_s,autorTXTP.text= $Autor[vAUX].to_s, editoraTXTP.text= $Editora[vAUX].to_s, anoTXTP.text = $Ano[vAUX].to_s

 tituTXTP.text = $Titulo[vAUX].to_s,autorTXTP.text= $Autor[vAUX].to_s, editoraTXTP.text= $Editora[vAUX].to_s, anoTXTP.text = $Ano[vAUX].to_s

 tituTXTP.text = $Titulo[vAUX].to_s,autorTXTP.text= $Autor[vAUX].to_s, editoraTXTP.text= $Editora[vAUX].to_s, anoTXTP.text = $Ano[vAUX].to_s



T, anoTXT)

nTXT.text="", editoraTXT.text="", anoTXT.text =""

```



Esse programa acima tem como objetivo a procura rápida e ágil de livros armazenados em uma biblioteca a partir da digitação de seu código, nome do autor, livro, editora e até mesmo ano de lançamento, para que isso fosse possível foram criadas várias condições "IF/ELSE".

Utilizaram-se parâmetros como o "require_relative"para fazer com que o programa principal recebesse as classes implementadoras da arquitetura POO.

- Classe Executar: composta por métodos e construtores como "DefSalvar" que salva os valores coletados pelo usuário em um vetor e o "DefProcurar" que procura os valores desejados;
- Classe Erro: onde se é chamada apenas quando o programa detecta que não foi digitado nenhum valor em uma ou nas duas textfields, apresentando assim uma menssagebox informando o erro;

Desta maneira os métodos são definidos no "limpar", que terá um “return” com todas as varias.text = "" para limpar o textbox, e nos "returns" utilizados no procurar e no limpar

para retornarem algo. Já os construtores são definidos no "salvar", pois este é apenas o armazenamento.

Além dessas classes também foi implementada a classe principal, que apresenta os objetos do formulário. Para cada uma dessas classes foi preciso criar um objeto para instancia-las, assim após o requeri é colocado $variavel = nome da class.new, como exemplo utilizado no programa:

$ProcurarClass = Exemplo.new

4.5. POO EM RUBY: OUTRAS FORMAS DE IMPLEMENTAÇÃO

Tal como já foi citado anteriormente, classe é uma maneira de definição de linguagem orientada a objeto, a qual pode ser definida por atributos e seu comportamento definido por métodos. Por isso, existem diferentes formas de implementar e escrever um código baseado em POO, já que a lógica de programação pode variar entre os profissionais. Um exemplo disso é a implementação das classes que já foram utilizadas neste capítulo. A seguir apresentaremos outra forma de “escrever” classes em Ruby.

4.5.1. OUTRA FORMA DE IMPLEMENTAÇÃO: ÁREA DO TRIÂNGULO

Nesta nova forma de demonstrar o cálculo da área de um triângulo é possível perceber a diferença entre as classes de erro apresentadas. Percebe-se que no primeiro exemplo citado aqui (imagem abaixo) a classe Erro apresenta apenas uma “Mensage Box” informando erro de inicialização de dados.

```
#Classe erro
class Erro < FXMainWindow
    def initialize(app)
        FXMessageBox.information(app, MBOX_OK,"ERRO!", "CAIXA DE TEXTO VAZIA... POR FAVOR, INSIRA ALGUM VALOR")
    end
  end

```

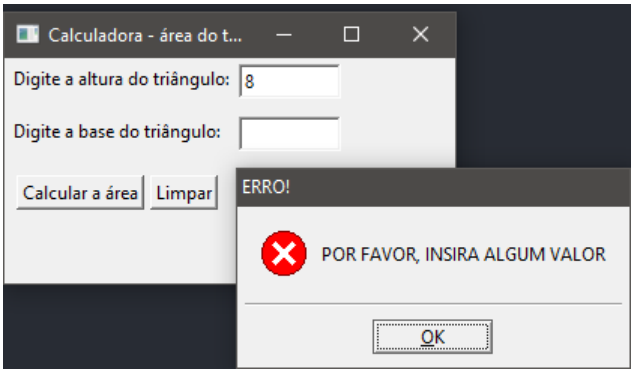
Essa classe pode ser feita de outra forma, tal como a imagem abaixo, onde, na própria Classe Erro, há a verificação da caixa de texto vazia, evitando a necessidade de conter o laço IF no ambiente ao qual ela foi chamada, ou seja, sempre que a Classe Erro for necessária existirá uma verificação de campos preenchidos. É possível perceber que ela é inicializada de forma diferente, quando usa os valores contidos nos campos com parâmetros iniciais (n1 e n2).

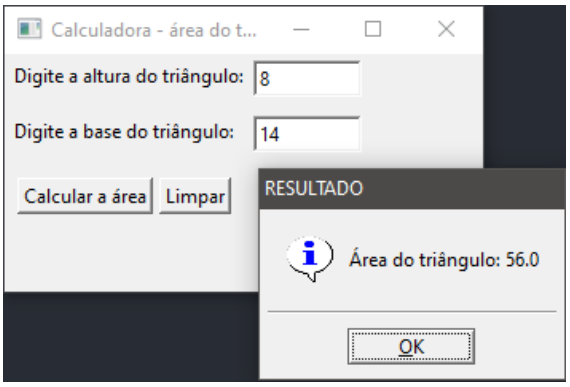
```
class Erro < FXMainWindow
  def initialize(app, n1, n2)
      #Condição para verificar se os campos estão vazios para chamar a classe CalcularArea
    if n1.text == "" || n2.text == ""
        FXMessageBox.error(app, MBOX_OK,"ERRO!", "POR FAVOR, INSIRA ALGUM VALOR!")
      else
        #Objeto de instância para ser utilizado na classe CalcularArea
        obj = CalcularArea.new(app, n1, n2)
    end
  end
end
```

Necessário ressaltar que essa classe é chamada através do botão Calcular presente na classe principal do programa.

```ruby
#Classe para limpar os campos
class Limpar < FXMainWindow
  def initialize(app, n1, n2)
    n1.text = ""
    n2.text = ""
  end
end

#Classe principal
class FXRuby < FXMainWindow
  def initialize(app)
    super(app, "Calculadora - área do triângulo", :width => 300, :height => 150)

    /Objetos do formulário/
    hFrame1 = FXHorizontalFrame.new(self)
    label = FXLabel.new(hFrame1, "Digite a altura do triângulo:")
    @valorAltura = FXTextField.new(hFrame1, 10)

    hFrame2 = FXHorizontalFrame.new(self)
    label = FXLabel.new(hFrame2, "Digite a base do triângulo:  ")
    @valorBase = FXTextField.new(hFrame2, 10)

    vFrame1 = FXVerticalFrame.new(self, :opts => LAYOUT_FILL)
    hFrame3 = FXHorizontalFrame.new(vFrame1)
    botaoCalcular = FXButton.new(hFrame3, "Calcular a área")
    botaoLimpar = FXButton.new(hFrame3, "Limpar")

```

```ruby
    #Objeto de instância para utilizar na classe Erro
    botaoCalcular.connect(SEL_COMMAND) do
         obj = Erro.new(app, @valorAltura, @valorBase)
    end

    botaoLimpar.connect(SEL_COMMAND) do
      obj = Limpar.new(app, @valorAltura, @valorBase)
    end
end


def create
  super
    show (PLACEMENT_SCREEN)
  end
end
#Chamando o Form
app = FXApp.new
#Chamando a Classe
FXRuby.new(app)
#Criando o Form
app.create
#Mantendo o Form aberto
app.run

```





4.5.2. OUTRA FORMA DE IMPLEMENTAÇÃO: SALÁRIO-HORISTA

Na classe de erro anteriormente citada, há novamente a utilização somente de uma MessageBox de aviso para campos de entrada de dados vazios, (imagem abaixo).

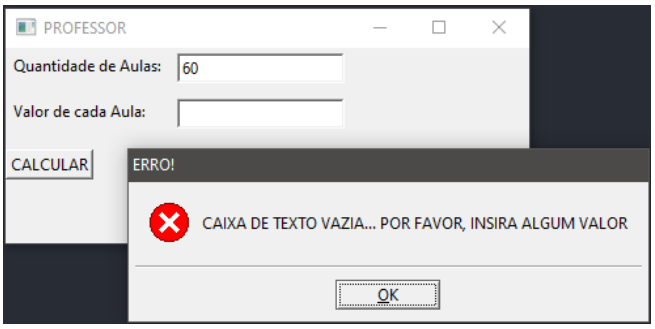
```ruby
#Classe Erro como caixa de mensagem
class Erro < FXMainWindow
    def initialize(app)
        FXMessageBox.information(app, MBOX_OK,"ERRO!", "CAIXA DE TEXTO VAZIA... POR FAVOR, INSIRA ALGUM VALOR")
    end
end
```

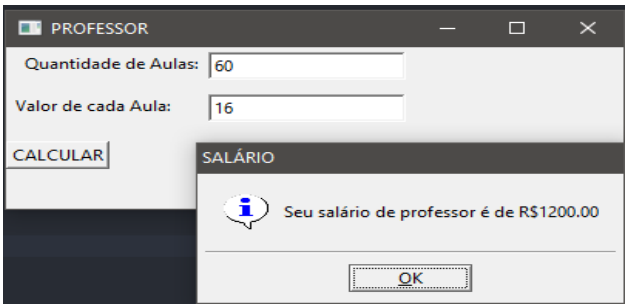
Entretanto, na nova modelação do script percebe-se que é possível colocar a condição (If/else) para a exibição da caixa de mensagem, tal como no exemplo da área do triângulo. Assim, quando a classe erro é chamada por meio do objeto presente em outras classes, ela já conterá a condição de existência, possibilitando a não necessidade de aplicar condições de existência para o objeto. (imagem abaixo).

```ruby
#Classe Erro como caixa de mensagem
class Erro < FXMainWindow
    def initialize(app, tipo, quantidade, valor)
        #Condição para verificar se os campos estão vazios para chamar a classe Resultado
        if quantidade.text == "" || valor.text == ""
          FXMessageBox.error(app, MBOX_OK,"ERRO!", "CAIXA DE TEXTO VAZIA... POR FAVOR, INSIRA ALGUM VALOR!")
          else
            #Objeto de instância para ser utilizado na classe Resultado
            obj = Resultado.new(app, tipo, quantidade, valor)
    end
 end
end
```

```ruby
#Classe Horista
class Salario_Horista < FXMainWindow
  def initialize(app)
   super(app, "HORISTA", :width => 400, :height => 150)

   frame = FXHorizontalFrame.new(self)
   frame2 = FXHorizontalFrame.new(self)

   quantidadeLabel = FXLabel.new(frame, "Quantidade de Aulas:  ")
   valorLabel = FXLabel.new(frame2, "Valor de cada Aula:      ")

   #Criando as caixas de texto que receberão os valores
   quantidade = FXTextField.new(frame, 20)
   valor = FXTextField.new(frame2, 20)

   salario = 0
   salarioLabel = FXLabel.new(self , "")

   button = FXButton.new(self, "CALCULAR")

   #Objeto de instância para utilizar na classe Erro
   button.connect(SEL_COMMAND) do
     obj = Erro.new(app, "horista", quantidade, valor)
   end
 end

  def create
    super
      show (PLACEMENT_SCREEN)
    end
```

Através da imagem acima, percebe-se o uso da classe de Erro, que é chamada por meio de um objeto presente na classe principal.





4.5.3. OUTRA FORMA DE IMPLEMENTAÇÃO: EQUAÇÃO DE SEGUNDO GRAU

Aqui foram feitas as mesmas modificações dos programas anteriores. No primeiro exemplo, a classe Erro apresentava apenas a mensageBox (imagem abaixo).

```
class Erro < FXMainWindow
  def initialize(app)
    FXMessageBox.error(app, MBOX_OK,"ERRO!", "Caixa de texto vazia")
  end
end
```

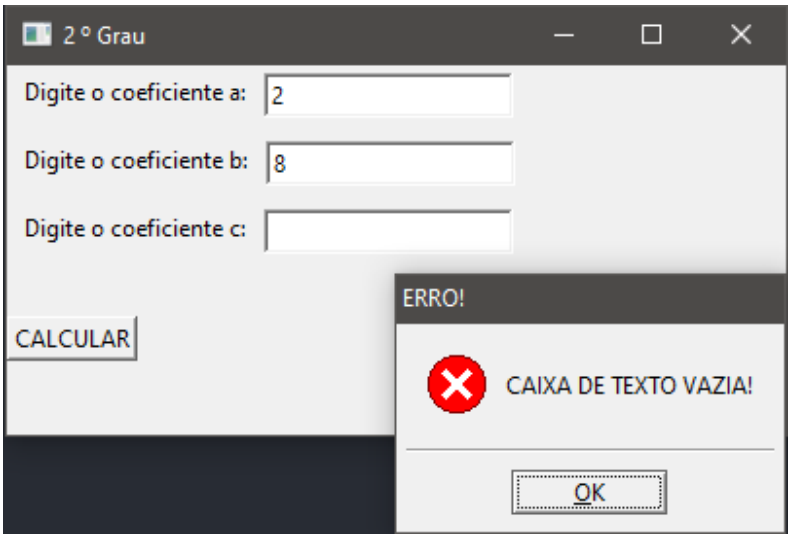
Já na forma atualizada, houve novos parâmetros de entrada (a, b, dn, raiz1, raiz2) que permitiram a utilização das condições de existência dentro da própria classe. (imagem abaixo).

```
#Classe Erro
class Erro < FXMainWindow
  def initialize(app, a, b, c, dn, raiz1, raiz2)
    #Condição para verificar se os campos estão vazios para chamar a classe Resultado
    if a.text == "" || b.text == "" || c.text == ""
      FXMessageBox.error(app, MBOX_OK,"ERRO!", "CAIXA DE TEXTO VAZIA!")
    else
      #Objeto de instância para ser utilizado na classe Resultado
      obj = Resultado.new(app, a, b, c, dn, raiz1, raiz2)
    end
  end
end
```

```
#Classe Resultado
class Resultado < FXMainWindow
  def initialize(app, a, b, c, dn, raiz1, raiz2)

    delta = (b.text.to_i * b.text.to_i) - ( 4 * a.text.to_i * c.text.to_i)

    if delta < 0
      dn.text = "Não existem raizes reais"
    else
      x1 = ((-b.text.to_i + Math.sqrt(delta) )/(2 *a.text.to_i))
      x2 = ((-b.text.to_i - Math.sqrt(delta) )/(2 *a.text.to_i))

      raiz1.text = "X1: #{x1}"
      raiz2.text = "X2: #{x2}"
  end
end
```

```ruby
end

#Classe principal
class Segundo < FXMainWindow
  def initialize(app)
    super(app, "2 º Grau", :width => 400, :height => 190)

    frame = FXHorizontalFrame.new(self)
    frame2 = FXHorizontalFrame.new(self)
    frame3 = FXHorizontalFrame.new(self)

    a = FXLabel.new(frame,  " Digite o coeficiente a: ")
    b = FXLabel.new(frame2, " Digite o coeficiente b: ")
    b = FXLabel.new(frame3, " Digite o coeficiente c: ")

    aa = FXTextField.new(frame, 20)
    bb = FXTextField.new(frame2, 20)
    cc = FXTextField.new(frame3, 20)

    delta = 0
    x1 = 0
    x2 = 0
    dn = FXLabel.new(self , "")
    raiz1 = FXLabel.new(self , "")
    raiz2 = FXLabel.new(self , "")
    button = FXButton.new(self, "CALCULAR")

    #Objeto de instância para utilizar na classe Erro
    button.connect(SEL_COMMAND) do
       obj = Erro.new(app, aa, bb, cc, dn, raiz1, raiz2)
    end
  end
```

```ruby

  def create
    super
      show (PLACEMENT_SCREEN)
    end
end

#Chamando o Form
app = FXApp.new
#Chamando a Classe
Segundo.new(app)
#Criando o Form
app.create
#Mantendo o Form aberto
app.run
```



4.5.4. OUTRA FORMA DE IMPLEMENTAÇÃO: CADASTRO DE LIVROS

Novamente vislumbra-se que a principal diferença entre as classes Erro apresentadas é a utilização de parâmetros de entrada que permitem o uso de laços de repetição, os quais, quando presentes na classe Erro (tal como na segunda imagem acima), permitem que a repetição ocorra sempre que a classe for solicitada, nesse caso, isso acontece quando o botão (o qual possui o objeto referente a classe) é acionado.

```ruby
class Erro < FXMainWindow
  def initialize(app)
    FXMessageBox.error(app, MBOX_OK,"ERRO!", "Tem um epaço em branco!! :(")
  end
end
```

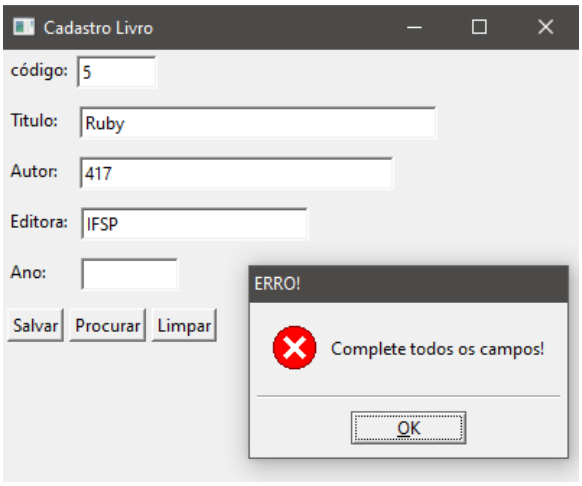
```ruby
require 'fox16'
include Fox
require_relative 'Codigo'

$ProcuarClass = Executar.new
$SalvarCLass = Executar.new
$LimparCLass = Executar.new
=begin
    Class para o visual
    classe para o cadastro
    class para a exclusão
    class para a pesquisa
=end

#Classe Erro
class Erro < FXMainWindow
  def initialize(app, codigo, titulo, autor, editora, ano, auxSalvar)
    #Condição para verificar se os campos estão vazios para chamar o método Salvar da classe Executar
    if codigo.text == "" || titulo.text == "" || autor.text == "" || editora.text == "" || ano.text == ""
      FXMessageBox.error(app, MBOX_OK,"ERRO!", "Complete todos os campos!")
    else
        #Objeto de instância para ser utilizado no método Salvar da classe Executar
        $SalvarCLass.Salvar(codigo, titulo, autor, editora, ano, auxSalvar)
        auxSalvar += 1
      end
    end
  end
```

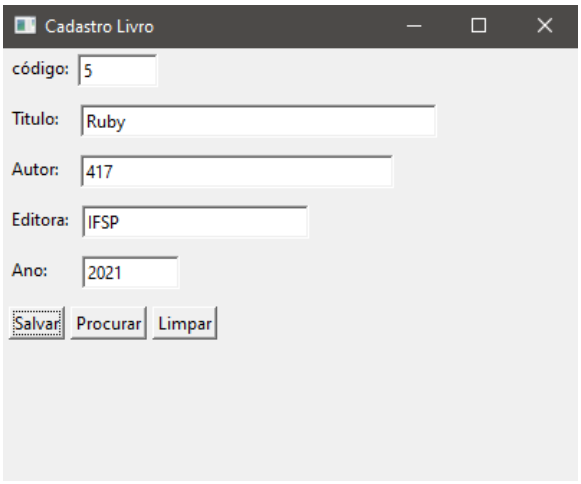
```ruby
#Classe principal
class Fxruby < FXMainWindow

    def initialize(app)
      super(app, "Cadastro Livro", :width => 400, :height => 300 )

      #crindo caixas "invisiveis" para uma melhor organização dos componentes
      hFrame1 = FXHorizontalFrame.new(self)
      hframe2 = FXHorizontalFrame.new(self)
      hFrame3 = FXHorizontalFrame.new(self)
      hFrame4 = FXHorizontalFrame.new(self)
      hFrame5 = FXHorizontalFrame.new(self)
      hFrame6 = FXHorizontalFrame.new(self)
      #esse é o text e o label do codigo
      codigoLabel = FXLabel.new(hFrame1, "código:")
      codigo = FXTextField.new(hFrame1, 8)
      #já esse é o text e o label do titulo
      tituloLabel = FXLabel.new(hframe2 , "Titulo:   ")
      titulo = FXTextField.new(hframe2, 40)
      #esse é o text e o label do autor
      autorLabel = FXLabel.new(hFrame3, "Autor:   ")
      autor = FXTextField.new(hFrame3, 35)
      #já esse é o text e o label da editora
      editoraLabel = FXLabel.new(hFrame4 , "Editora: " )
      editora = FXTextField.new(hFrame4, 25)
      #já esse é o text e o label do ano
      anoLabel = FXLabel.new(hFrame5 , "Ano:      " )
      ano = FXTextField.new(hFrame5, 10)

      #os botões
      btnSalvar = FXButton.new(hFrame6, "Salvar" )
      btnPreocurar = FXButton.new(hFrame6, "Procurar" )
      btnLimpar = FXButton.new(hFrame6, "Limpar" )
```

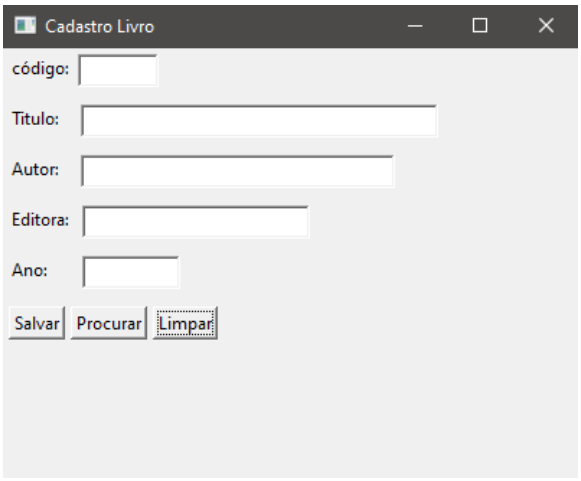
```
      auxProcura = 1
      btnPreocurar.connect(SEL_COMMAND) do
        #Objeto de instância para utilizar no método Procurar da classe Executar
        $ProcuarClass.Procurar(codigo, titulo, autor, editora, ano, auxProcura)
        auxProcura += 1
      end

      btnLimpar.connect(SEL_COMMAND) do
        #Objeto de instância para utilizar no método Limpar da classe Executar
        $LimparClass.Limpar(codigo, titulo, autor, editora, ano)
      end

      auxSalvar = 1
      btnSalvar.connect(SEL_COMMAND) do
        #Objeto de instância para utilizar na classe Erro
        obj = Erro.new(app, codigo, titulo, autor, editora, ano, auxSalvar)
      end
    end
    def create
      super
      show (PLACEMENT_SCREEN)
    end
end
app = FXApp.new
#chamando a classe FXruby
Fxruby.new(app)
#criando o forme
app.create
#mantendo o forme
app.run
```

```
require 'fox16'
include Fox

class Executar

  #colocando os vetores em global
  $Codigo = Array.new(10)
  $Titulo = Array.new(10)
  $Autor = Array.new(10)
  $Editora = Array.new(10)
  $Ano = Array.new(10)

  def Salvar(codLivro, tituLivro, autorLivro, editoLivro, anoLivro, aux)
    #armazenando os valores
    $Codigo[aux] = codLivro
    $Titulo[aux] = tituLivro
    $Autor[aux] = autorLivro
    $Editora[aux] = editoLivro
    $Ano[aux] = anoLivro
    puts("#{$Codigo[aux]}\n#{$Titulo[aux]}\n#{$Autor[aux]}\n#{$Editora[aux]}\n#{$Ano[aux]}")
  end

```

```
  def Procurar(codTXTP, tituTXTP, autorTXTP, editoraTXTP, anoTXTP, vAUX)
    while vAUX < 10

      #verificando cada variavel caso a pessoa tenha colocado para buscar sem ser pelo o codigo
      if $Codigo[vAUX] == codTXTP
        return codTXTP.text = $Codigo[vAUX].to_s, tituTXTP.text = $Titulo[vAUX].to_s,autorTXTP.text= $Autor[vAUX].to_s, editoraTXTP.text= $Editora[vAUX].to_s, anoTXTP.text = $Ano[vAUX].to_s
      elsif $Titulo[vAUX] == codTXTP
          return codTXTP.text = $Codigo[vAUX].to_s, tituTXTP.text = $Titulo[vAUX].to_s,autorTXTP.text= $Autor[vAUX].to_s, editoraTXTP.text= $Editora[vAUX].to_s, anoTXTP.text = $Ano[vAUX].to_s
      elsif $Autor[vAUX] == codTXTP
        return codTXTP.text = $Codigo[vAUX].to_s, tituTXTP.text = $Titulo[vAUX].to_s,autorTXTP.text= $Autor[vAUX].to_s, editoraTXTP.text= $Editora[vAUX].to_s, anoTXTP.text = $Ano[vAUX].to_s
      elsif $Editora[vAUX] == codTXTP
        return codTXTP.text = $Codigo[vAUX].to_s, tituTXTP.text = $Titulo[vAUX].to_s,autorTXTP.text= $Autor[vAUX].to_s, editoraTXTP.text= $Editora[vAUX].to_s, anoTXTP.text = $Ano[vAUX].to_s
      elsif $Ano[vAUX] == codTXTP
        return codTXTP.text = $Codigo[vAUX].to_s, tituTXTP.text = $Titulo[vAUX].to_s,autorTXTP.text= $Autor[vAUX].to_s, editoraTXTP.text= $Editora[vAUX].to_s, anoTXTP.text = $Ano[vAUX].to_s
      end
    end
  end

  def Limpar(codTXT, tituTXT, autorTXT, editoraTXT, anoTXT)
    #limpando os textbox
    return codTXT.text="", tituTXT.text ="", autorTXT.text="", editoraTXT.text="", anoTXT.text =""
  end
end
```


Cadastro Livro
código: 5
Titulo: Ruby
Autor: 417
Editora: IFSP
Ano:
Salvar
Procurar
Limpar
ERRO!
Complete todos os campos!
OK


Cadastro Livro
código: 5
Titulo: Ruby
Autor: 417
Editora: IFSP
Ano: 2021
Salvar
Procurar
Limpar


Cadastro Livro
código:
Titulo:
Autor:
Editora:
Ano:
Salvar
Procurar
Limpar

5. CAPÍTULO: RUBY E BANCO DE DADOS

5.1. BANCO DE DADOS

Banco de dados é uma coleção organizada de informações ou dados, ou seja, são informações ou dados interligados, sempre relacionados a alguém ou a algo. Serve justamente para coletar e armazenar dados e informações de forma segura, organizada e padronizada.

Foi criado no ano de 1960, desde então, evoluiu com o passar do tempo. Os bancos de dados hierárquicos de redes eram sistemas originais usados para armazenar e manipular dados. Em 1980 os bancos de dados relacionais se tornaram bem populares, em sequência o banco de dados orientado a objetos em 1990. O mais recente é o banco de dados NoSQL, que surgiu como uma espécie de resposta ao crescimento da internet e com às necessidades de maior velocidade e processamento de dados não estruturados.

Atualmente, os bancos de dados na nuvem e autônomos estão abrindo novos seguimentos quando se trata de como os dados são coletados, armazenados, gerenciados e utilizados.

O banco de dados é controlado pelo DBMS: sistema de gerenciamento de banco de dados. O DBMS funciona como uma espécie de interface entre o banco de dados e seus usuários finais ou programas, onde permite que os usuários recuperem, atualizem e gerenciem como serão organizadas e otimizadas todas as informações que serão coletadas. Ademais, também permite uma variedade de operações administrativas, como, por exemplo, o monitoramento de desempenho, os ajustes e os backups e recuperações.

O BDD (Banco de Dados) mais recente era o relacionado, mas foi criada uma alternativa para esse modelo que é chamado de: banco de dados não-relacionado. A diferença entre os dois é que o relacionado é organizado e visualizado em forma de tabelas, já o não-relacionado é organizado por chaves, ou seja, cada chave é utilizada para uma recuperação de um valor, com a visualização diferente da tradicional.

Nos dias atuais, a utilização de BDD é muito necessária na vida industrial, pois tem várias vantagens que melhoram a produção e o controle das indústrias e empresas no geral. Como por exemplo:

- A produtividade;
- O aumento da segurança;
- Melhor relacionamento;
- Melhora do planejamento e decisões;
- Redução de riscos.

No entanto, mesmo com esses benefícios, enfrentam grandes desafios, já que os bancos de dados comerciais utilizam consultas muito complexas obtendo respostas instantâneas a essas. Logo, os administradores precisam empregar uma ampla variedade de métodos para melhorar o desempenho, mas, para essa finalidade, enfrentam algumas adversidades como:

- Absorção de aumentos significativos no volume de dados;
- Garantia da segurança de dados;
- Acompanhamento da demanda;
- Gerenciamento e manutenção do banco de dados e da infra-estrutura;
- Remoção de limites na escalabilidade;
- Por ter muitos dados, a manutenção pode demorar mais e, portanto, prejudicar a empresa.

5.2. O SQLITE

O SQLite é um banco de dados relacional que não armazena informações em um servidor, diferente das outras ferramentas do tipo. Sua independência é devido a capacidade de armazenar arquivos dentro de si mesmo; ele lê e grava diretamente em arquivos de disco comuns.

O projeto SQLite foi iniciado em 09/05/2000 . O futuro é sempre difícil de prever, mas a intenção dos desenvolvedores é oferecer suporte ao SQLite até o ano 2050. As decisões de design são feitas com esse objetivo em mente.

Considerado um banco de dados embutido, o SQLite é completo com várias tabelas, índices, gatilhos e visualizações e está contido em um único arquivo de disco. O formato do arquivo de banco de dados é multi-plataforma (pode copiar livremente um banco de dados entre sistemas de 32 e 64 bits). Esses recursos tornam o SQLite uma escolha popular como formato de arquivo de aplicativo, tanto

que seus arquivos de banco de dados são um formato de armazenamento recomendado pela Biblioteca do Congresso dos EUA.

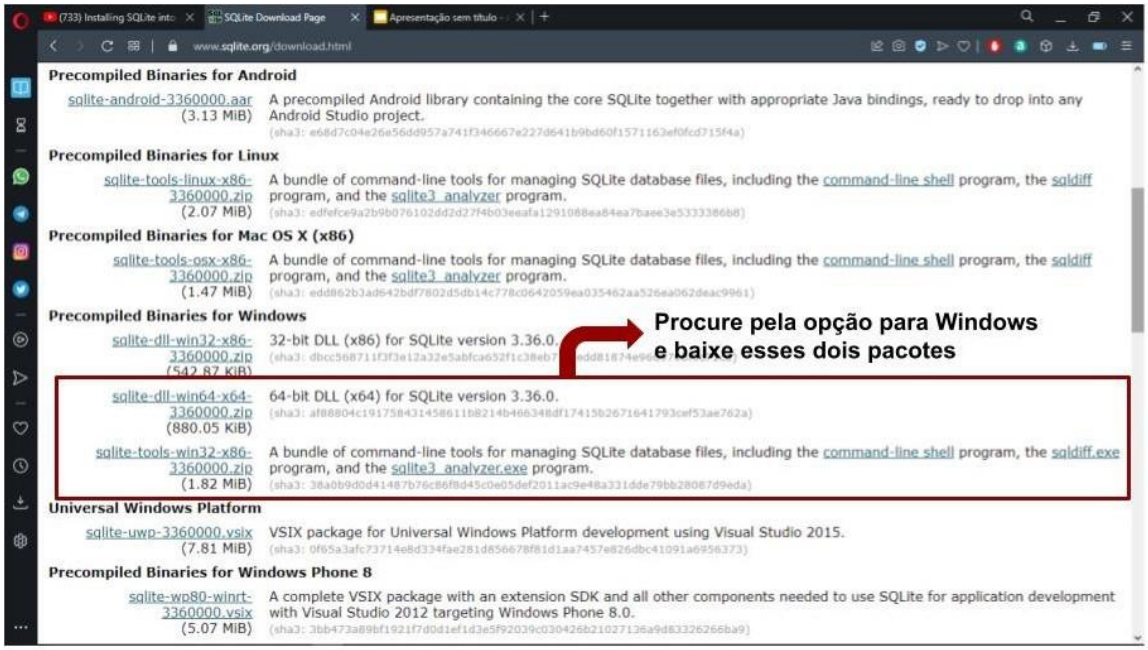
Essa base de dados é de código aberto e gratuito, sendo muito utilizada em aplicações *mobile*, com foco no sistema Android.

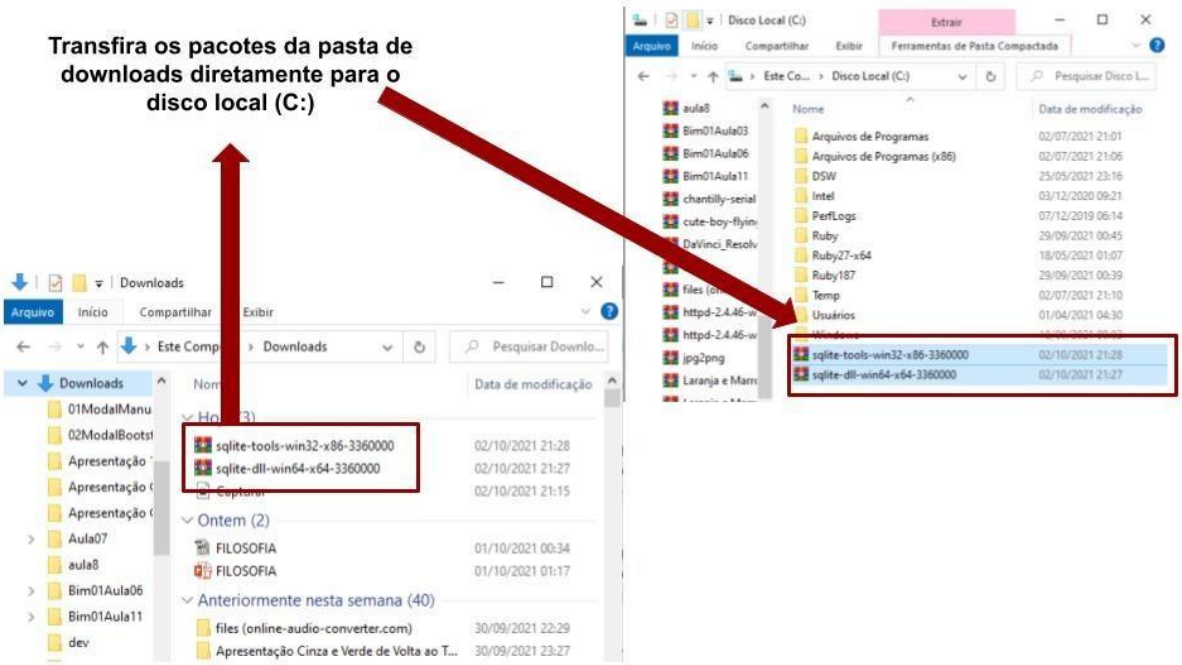
O SQLite, com suas características e funcionalidades específicas, coloca-se como uma opção de banco de dados relevante para algumas situações. Como, por exemplo:

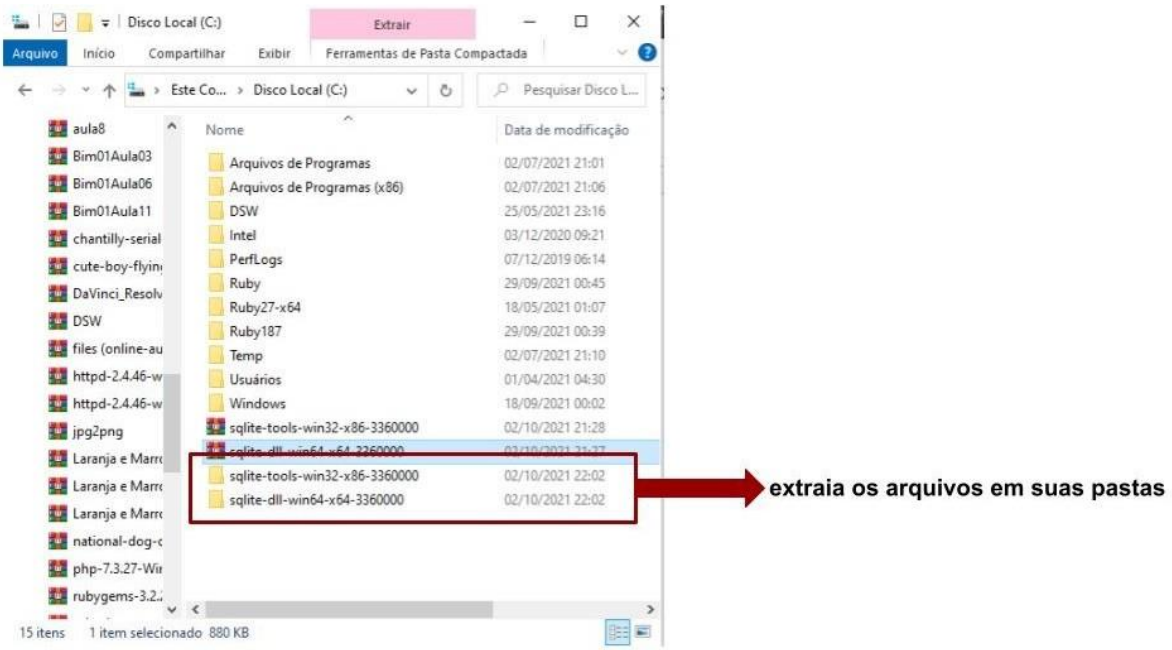
- Sites que tenham um tráfego médio, especialmente falando de acessos simultâneos. Essa costuma ser a média, exceto grandes e-commerces, ou em períodos como Black Friday;
- Dispositivos que funcionam sem a necessidade de que alguém os opere. O encaixe é perfeito com o SQLite pelo fato de ele trabalhar também de forma independente, sem que alguém precise administrá-lo. Por isso, entre os dispositivos ideais para ele: smartphones, TVs, relógios, eletrodomésticos inteligentes, câmeras, consoles de videogame, etc;
- Aplicações desktop diversas, como: sistemas financeiros, de edição de mídia, controle de versão e programas de manutenção de registros, por exemplo. Quando usado dessa maneira, o SQLite possibilita um desempenho muito maior nessas aplicações, o que garante maior certeza de um funcionamento estável;
- O SQLite consegue interagir com aplicativos de qualquer linguagem, desde que as suas bibliotecas externas tenham sido escritas em C. Dessa forma, o processo é mais simples, principalmente pela dispensa da relação usuário-servidor

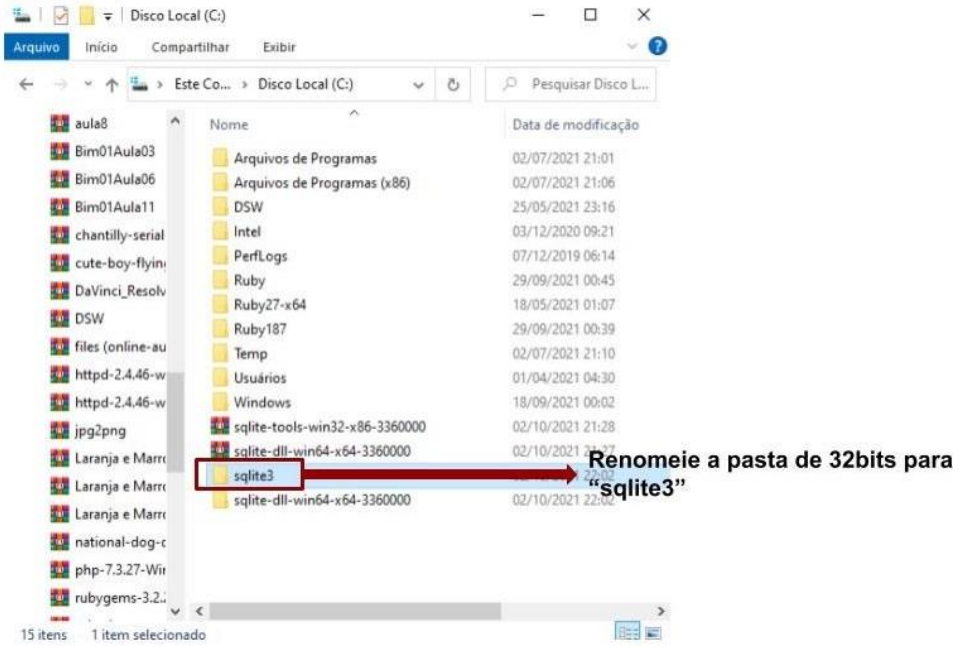
Hoje, graças a todo seu potencial fluido e dinâmico, o SQLite ocupa a 9ª opção de base de dados mais utilizada em todo o mundo.

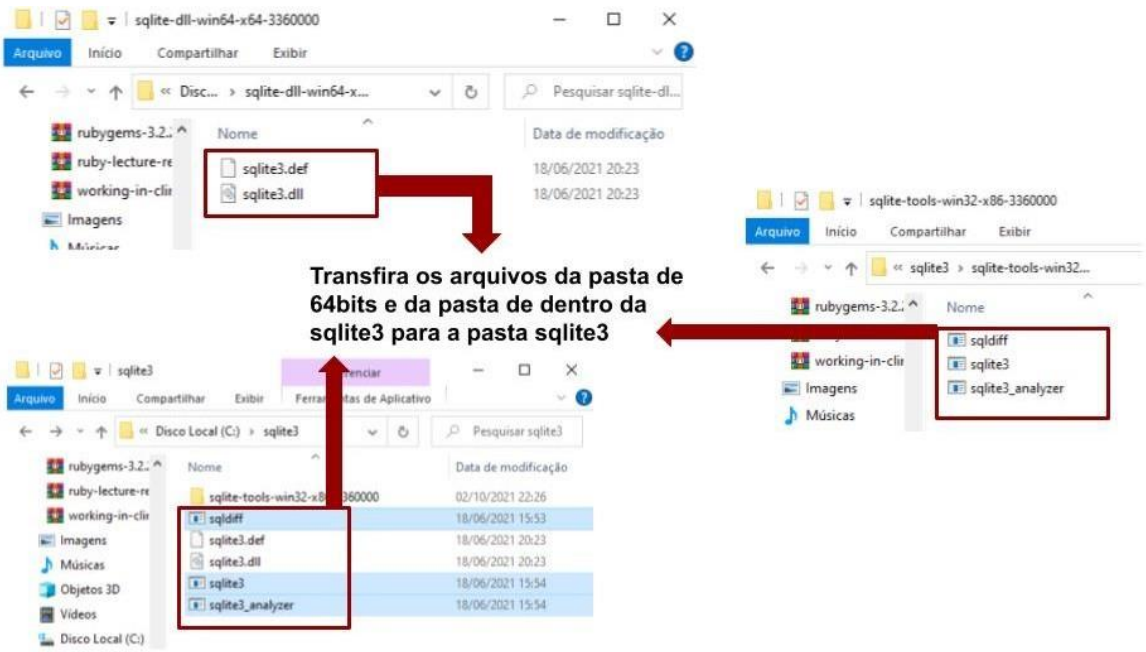
5.2.1. SQLITE: INSTALAÇÃO

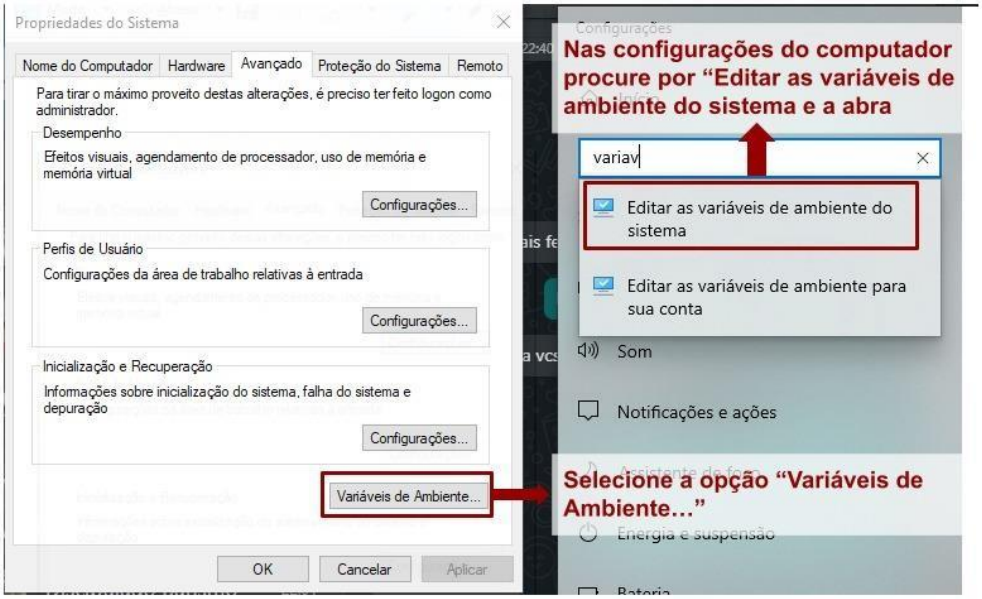

Precompiled Binaries for Android
sqlite-android-3360000.aar (3.13 MiB) A precompiled Android library containing the core SQLite together with appropriate Java bindings, ready to drop into any Android Studio project.
Precompiled Binaries for Linux
sqlite-tools-linux-x86-3360000.zip (2.07 MiB) A bundle of command-line tools for managing SQLite database files, including the command-line shell program, the sqldiff program, and the sqlite3_analyzer program.
Precompiled Binaries for Mac OS X (x86)
sqlite-tools-osx-x86-3360000.zip (1.47 MiB) A bundle of command-line tools for managing SQLite database files, including the command-line shell program, the sqldiff program, and the sqlite3_analyzer program.
Precompiled Binaries for Windows
Procure pela opção para Windows e baixe esses dois pacotes
sqlite-dll-win32-x86-3360000.zip (542.87 KiB) 32-bit DLL (x86) for SQLite version 3.36.0.
sqlite-dll-win64-x64-3360000.zip (880.05 KiB) 64-bit DLL (x64) for SQLite version 3.36.0.
sqlite-tools-win32-x86-3360000.zip (1.82 MiB) A bundle of command-line tools for managing SQLite database files, including the command-line shell program, the sqldiff.exe program, and the sqlite3_analyzer.exe program.
Universal Windows Platform
sqlite-uwp-3360000.vsix (7.81 MiB) VSIX package for Universal Windows Platform development using Visual Studio 2015.
Precompiled Binaries for Windows Phone 8
sqlite-wp80-winrt-3360000.vsix (5.07 MiB) A complete VSIX package with an extension SDK and all other components needed to use SQLite for application development with Visual Studio 2012 targeting Windows Phone 8.0.

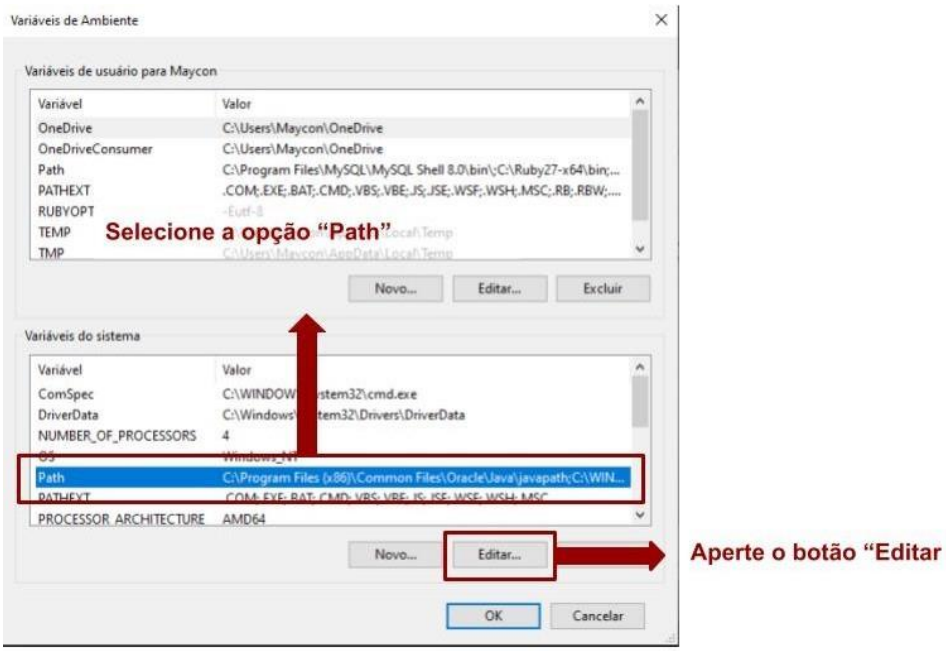

Transfira os pacotes da pasta de downloads diretamente para o disco local (C:)
Downloads
Arquivo
Início
Compartilhar
Exibir
Este Computador
Downloads
Pesquisar Downloads
Data de modificação
sqlite-tools-win32-x86-3360000
sqlite-dll-win64-x64-3360000
02/10/2021 21:28
02/10/2021 21:27
02/10/2021 21:15
Ontem (2)
FILOSOFIA
01/10/2021 00:34
FILOSOFIA
01/10/2021 01:17
Anteriormente nesta semana (40)
files (online-audio-converter.com)
30/09/2021 22:29
Apresentação Cinza e Verde de Volta ao T...
30/09/2021 23:27
Disco Local (C:)
Extrair
Ferramentas de Pasta Compactada
Nome
Arquivos de Programas
Arquivos de Programas (x86)
DSW
Intel
PerfLogs
Ruby
Ruby27-x64
Ruby187
Temp
Usuários
sqlite-tools-win32-x86-3360000
sqlite-dll-win64-x64-3360000


Disco Local (C:)
Extrair
Arquivo
Início
Compartilhar
Exibir
Ferramentas de Pasta Compactada
Este Co... › Disco Local (C:)
Pesquisar Disco L...
Nome
Data de modificação
Arquivos de Programas 02/07/2021 21:01
Arquivos de Programas (x86) 02/07/2021 21:06
DSW 25/05/2021 23:16
Intel 03/12/2020 09:21
PerfLogs 07/12/2019 06:14
Ruby 29/09/2021 00:45
Ruby27-x64 18/05/2021 01:07
Ruby187 29/09/2021 00:39
Temp 02/07/2021 21:10
Usuários 01/04/2021 04:30
Windows 18/09/2021 00:02
sqlite-tools-win32-x86-3360000 02/10/2021 21:28
sqlite-tools-win32-x86-3360000 02/10/2021 22:02
sqlite-dll-win64-x64-3360000 02/10/2021 22:02
15 itens
1 item selecionado 880 KB
extraia os arquivos em suas pastas

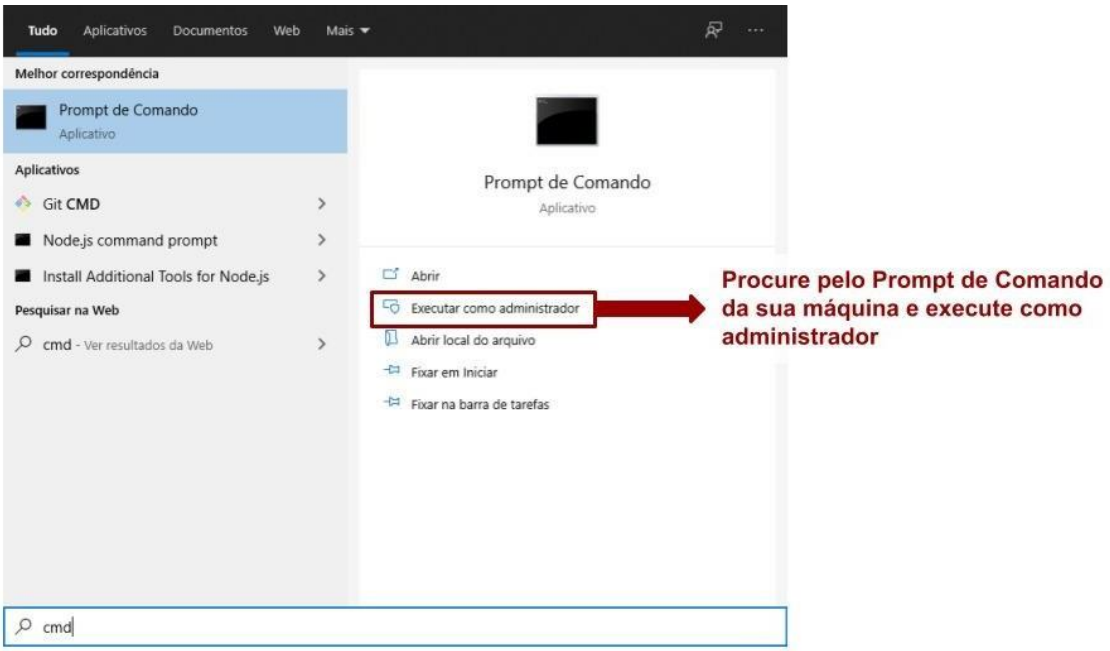

Disco Local (C:)
Arquivo
Início
Compartilhar
Exibir
Este Co... › Disco Local (C:)
Pesquisar Disco L...
Nome
Data de modificação
Arquivos de Programas 02/07/2021 21:01
Arquivos de Programas (x86) 02/07/2021 21:06
DSW 25/05/2021 23:16
Intel 03/12/2020 09:21
PerfLogs 07/12/2019 06:14
Ruby 29/09/2021 00:45
Ruby27-x64 18/05/2021 01:07
Ruby187 29/09/2021 00:39
Temp 02/07/2021 21:10
Usuários 01/04/2021 04:30
Windows 18/09/2021 00:02
sqlite-tools-win32-x86-3360000 02/10/2021 21:28
sqlite-dll-win64-x64-3360000
sqlite3
sqlite-dll-win64-x64-3360000 02/10/2021 22:02
15 itens
1 item selecionado
Renomeie a pasta de 32bits para "sqlite3"

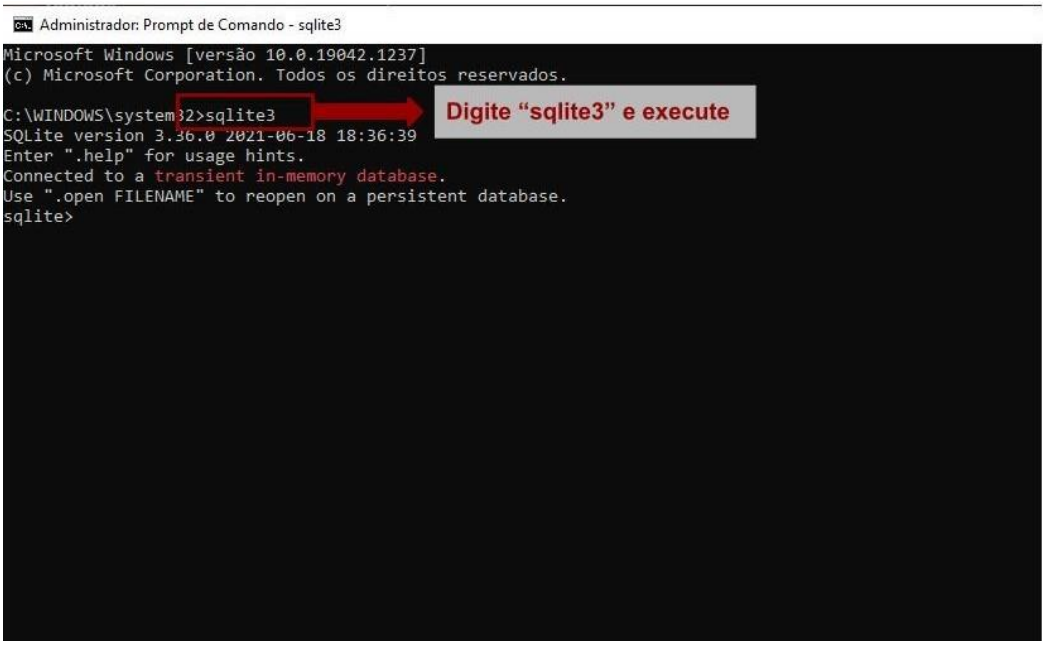

sqlite-dll-win64-x64-3360000
sqlite3.def
sqlite3.dll
Transfira os arquivos da pasta de 64bits e da pasta de dentro da sqlite3 para a pasta sqlite3
sqlite-tools-win32-x86-3360000
sqldiff
sqlite3
sqlite3_analyzer
sqlite3


Propriedades do Sistema
Nome do Computador
Hardware
Avançado
Proteção do Sistema
Remoto
Variáveis de Ambiente...
OK
Cancelar
Aplicar
Nas configurações do computador procure por "Editar as variáveis de ambiente do sistema e a abra
variav
Editar as variáveis de ambiente do sistema
Editar as variáveis de ambiente para sua conta
Som
Notificações e ações
Selecione a opção "Variáveis de Ambiente..."

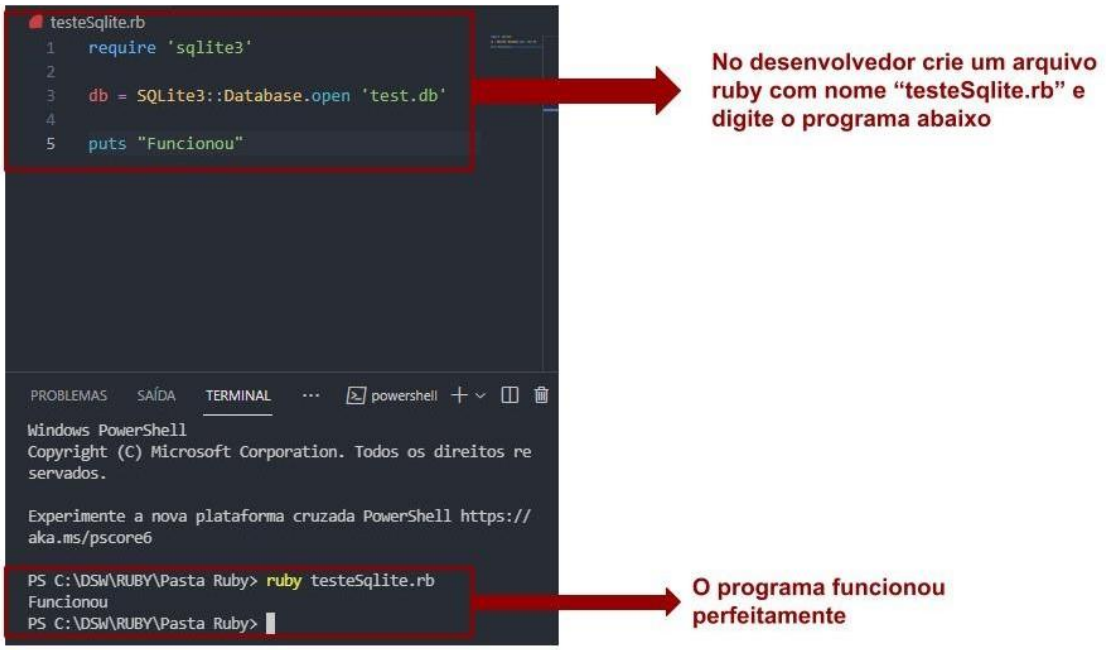

Variáveis de Ambiente
Variáveis de usuário para Maycon
Variável
Valor
OneDrive
C:\Users\Maycon\OneDrive
OneDriveConsumer
C:\Users\Maycon\OneDrive
Path
C:\Program Files\MySQL\MySQL Shell 8.0\bin\;C:\Ruby27-x64\bin;...
PATHEXT
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.RB;.RBW;....
RUBYOPT
TEMP
TMP
Selecione a opção "Path"
Novo...
Editar...
Excluir
Variáveis do sistema
Variável
Valor
ComSpec
DriverData
NUMBER_OF_PROCESSORS
4
Path
C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\WIN...
PROCESSOR_ARCHITECTURE
AMD64
Novo...
Editar...
Aperte o botão "Editar
OK
Cancelar


Editar a variável de ambiente
Aperte o botão "Novo"
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Git\cmd
C:\Program Files (x86)\nodejs\
C:\sqlite3
Cole o caminho da pasta "sqlite3" e aperte "ok" e todas as opções.
Novo
Editar
Procurar...
Excluir
Mover para baixo
Editar texto...
OK
Cancelar


Tudo
Aplicativos
Documentos
Web
Mais
Melhor correspondência
Prompt de Comando
Aplicativo
Aplicativos
Git CMD
Node.js command prompt
Install Additional Tools for Node.js
Pesquisar na Web
cmd - Ver resultados da Web
Prompt de Comando
Aplicativo
Abrir
Executar como administrador
Abrir local do arquivo
Fixar em Iniciar
Fixar na barra de tarefas
cmd
Procure pelo Prompt de Comando da sua máquina e execute como administrador


Administrador: Prompt de Comando - sqlite3
Microsoft Windows [versão 10.0.19042.1237]
(c) Microsoft Corporation. Todos os direitos reservados.
C:\WINDOWS\system32>sqlite3
SQLite version 3.36.0 2021-06-18 18:36:39
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
Digite “sqlite3” e execute


Administrador: Prompt de Comando
Microsoft Windows [versão 10.0.19042.1237]
(c) Microsoft Corporation. Todos os direitos reservados.
C:\WINDOWS\system32>sqlite3
SQLite version 3.36.0 2021-06-18 18:36:39
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> create database banco
...> ;
sqlite> .quit
C:\WINDOWS\system32>gem install sqlite3
Using msys2 packages: mingw-w64-x86_64-sqlite3
Building native extensions. This could take a while...
Done installing documentation for
1 gem installed
C:\WINDOWS\system32>
Digite ".quit" para sair do "sqlite3"
Digite "gem install sqlite3" para baixar a gema necessária para fazer a conexão entre o Ruby e o SQLite


testeSqlite.rb
require 'sqlite3'
db = SQLite3::Database.open 'test.db'
puts "Funcionou"
No desenvolvedor crie um arquivo ruby com nome "testeSqlite.rb" e digite o programa abaixo
PROBLEMAS
SAÍDA
TERMINAL
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.
Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6
PS C:\DSW\RUBY\Pasta Ruby> ruby testeSqlite.rb
Funcionou
PS C:\DSW\RUBY\Pasta Ruby>
O programa funcionou perfeitamente

5.3 PROGRAMA E BANCO DE DADOS

Agora que já se sabe o que é um banco de dados e como conectá-lo previamente a um programa Ruby, iremos criar um banco de dados SQLite para um programa já citado nos capítulos anteriores (programa de livros) com o intuito de maior fixação da relação BDD-Ruby.

Seguindo esse raciocínio, no programa abaixo, faz-se o cadastramento dos livros, com suas respectivas informações, ou seja, o BDD armazena os dados. Assim, quando deseja se fazer uma consulta em relação a um livro X, o programa irá procurá-lo nos dados armazenados do banco de dados e, por fim, retornará as informações ao usuário.

Logo, esse programa tem como objetivo a procura rápida e ágil de livros armazenados em uma biblioteca a partir da digitação de seu código, nome do autor, livro, editoria e até mesmo o ano de lançamento. Para isso, além da utilização de parâmetros, métodos e instâncias de classes, o programa foi conectado ao banco de dados

Por fim, os códidos abaixo foram utillizados para a criação do programa em questão.

- Arquivo relacionado ao livro:

oficial > Livro.rb

```ruby
require 'fox16'
include Fox
require_relative 'Codigo'
require 'sqlite3'

$db = SQLite3::Database.open 'Livraria.db'
$Livraria = Executar.new

class Erro < FXMainWindow
    def initialize(app, banco, codigo, titulo, autor, editora, ano)
      if codigo.text ==""|| titulo.text == ""|| autor.text ==""|| editora.text==""||ano.text==""
        FXMessageBox.error(app, MBOX_OK, "ERRO!", "Complete todos os campos!")
      else
        $Livraria.Salvar(banco, codigo.text, titulo.text, autor.text, editora.text, ano.text)
      end
    end
end

class Fxruby < FXMainWindow
    def initialize(app)
      super(app, "Cadastro Livro", :width => 400, :height => 300 )
      #crindo caixas "invisiveis" para uma melhor organização dos componentes
      hFrame1 = FXHorizontalFrame.new(self)
      hframe2 = FXHorizontalFrame.new(self)
      hFrame3 = FXHorizontalFrame.new(self)
      hFrame4 = FXHorizontalFrame.new(self)
      hFrame5 = FXHorizontalFrame.new(self)
      hFrame6 = FXHorizontalFrame.new(self)
      hFrame7 = FXHorizontalFrame.new(self)
```

```ruby
!igo, titulo, autor, editora, ano) VALUES(?,?,?,?,?)",[codLivro, tituLivro, autorLivro, editoLivro, anoLivro])



ntes ao codigo inserido, retornando em vetor
'SELECT * FROM Livro WHERE codigo = #{codTXTP}")


, editoraTXT, anoTXT)

xt ="", autorTXT.text="", editoraTXT.text="", anoTXT.text =""
```

```ruby
oficial > Livro.rb
54          obj = Erro.new(app, $db, codigo, titulo, autor, editora, ano)
55        end
56
57        btnPreocurar.connect(SEL_COMMAND) do
58        #Passando os valores que foram pegos no banco de dados
59          aux = $Livraria.Procurar($db, codigo)
60          codigo.text = String(aux[0])
61          titulo.text = String(aux[1])
62          autor.text = String(aux[2])
63          editora.text = String(aux[3])
64          ano.text = String(aux[4])
65        end
66
67        btnLimpar.connect(SEL_COMMAND) do
68          #limpando os campos dos valores
69          $Livraria.Limpar(codigo, titulo, autor, editora, ano)
70        end
71
72      end
73      def create
74        super
75        show (PLACEMENT_SCREEN)
76      end
77  end
78
79
80
81  app = FXApp.new
82  #chamando a classe FXruby
83  Fxruby.new(app)
84  #criando o forme
85  app.create
86  #mantendo o forme
87  app.run
```

- Parte relacionada ao código:

```ruby
oficial > Codigo.rb
1  require 'sqlite3'
2
3
4  class Executar
5      def Salvar(db, codLivro, tituLivro, autorLivro, editoLivro, anoLivro)
6          #armazenando os valores
7          db.execute("INSERT INTO Livro(codigo, titulo, autor, editora, ano) VALUES(?,?,?,?,?)",[codLivro,
8      end
9
10     def Procurar(db,codTXTP)
11         #Vai pegar os valores correspondentes ao codigo inserido, retornando em vetor
12         return valor = db.get_first_row("SELECT * FROM Livro WHERE codigo = #{codTXTP}")
13     end
14
15     def Limpar(codTXT, tituTXT, autorTXT, editoraTXT, anoTXT)
16         #limpando os textbox
17         return codTXT.text="", tituTXT.text ="", autorTXT.text="", editoraTXT.text="", anoTXT.text =""
18     end
19 end
```

```ruby
oficial > Codigo.rb
2
3
4
5   utorLivro, editoLivro, anoLivro)
6
7   digo, titulo, autor, editora, ano) VALUES(?,?,?,?,?)",[codLivro, tituLivro, autorLivro, editoLivro, anoLivro])
8
9
10
11  ntes ao codigo inserido, retornando em vetor
12  'SELECT * FROM Livro WHERE codigo = #{codTXTP}")
13
14
15  , editoraTXT, anoTXT)
16
17  xt ="", autorTXT.text="", editoraTXT.text="", anoTXT.text =""
18
19
```

- Parte reacionada ao teste:

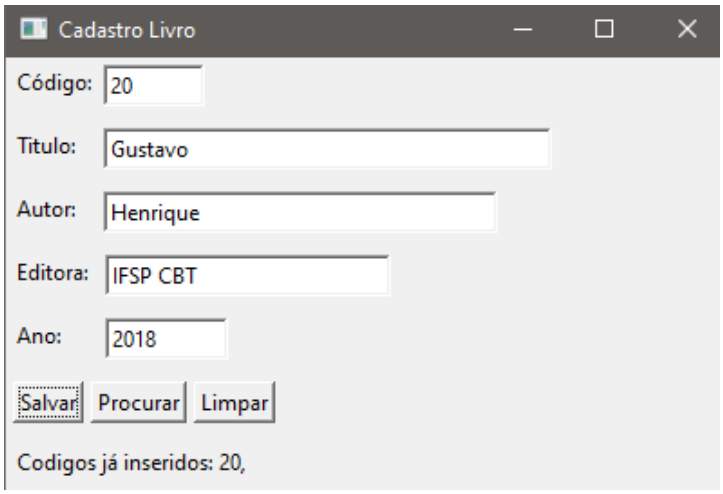
oficial > teste.rb

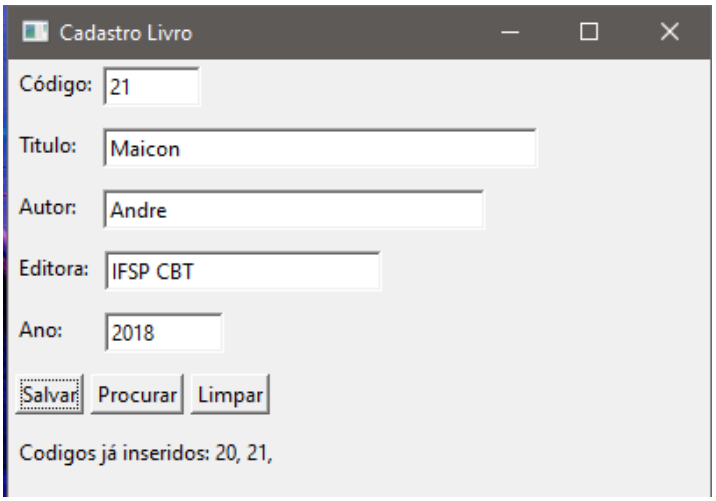
```ruby
require 'sqlite3'

#Fazendo a ligação com a database e criando uma nova database Livraria
db = SQLite3::Database.new 'Livraria.db'

#criando a tabela Livro
rows = db.execute <<-SQL
  create table Livro (
    codigo varchar(100),
    titulo varchar(100),
    autor varchar(100),
    editora varchar(100),
    ano varchar(100)
  );
SQL

```

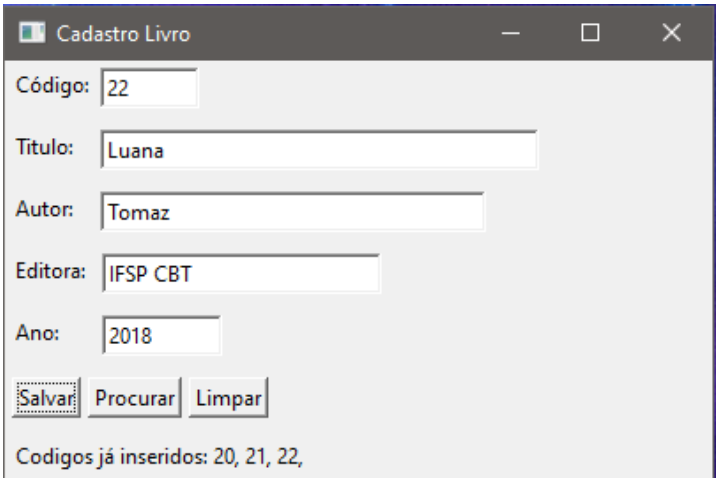
- Executando o banco de dados:

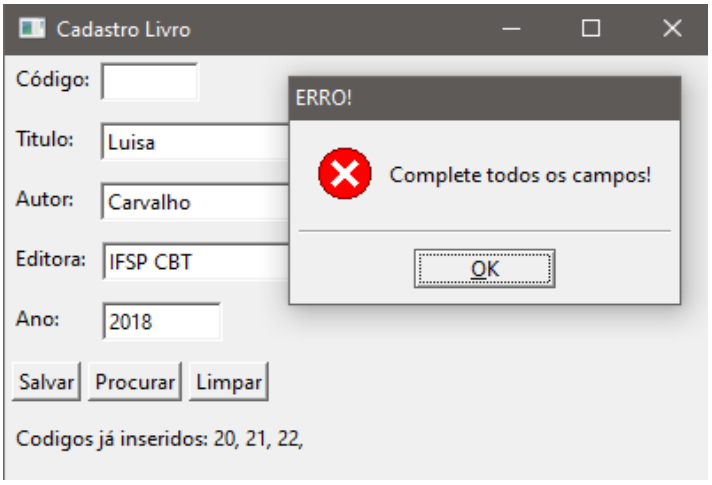
```
PS C:\xampp\htdocs\Testes em PHP\Ruby\oficial> ruby teste.rb
PS C:\xampp\htdocs\Testes em PHP\Ruby\oficial>
```

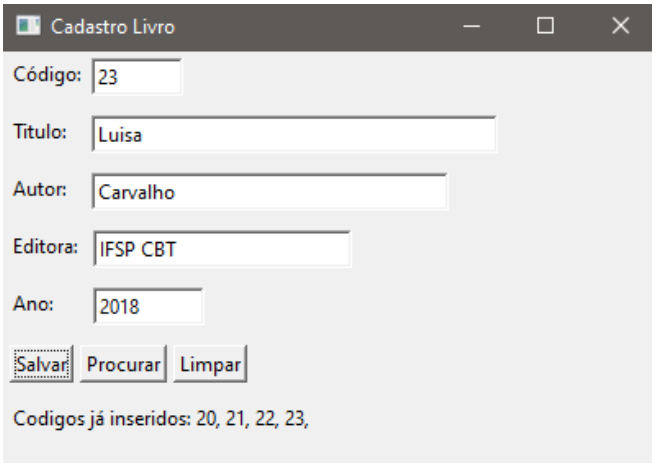
- Testando o cadastramento:





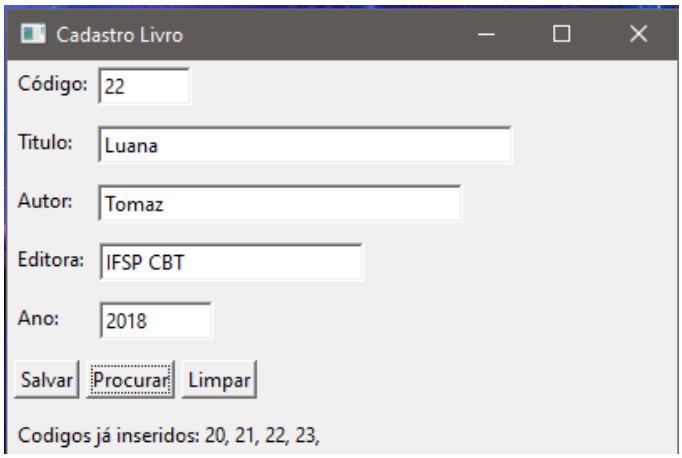


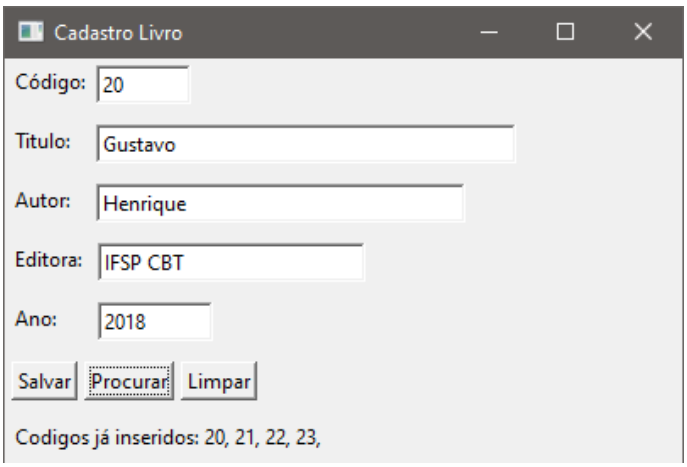




- Botão limpar:

- Pesquisando dados





6. CAPÍTULO: INTEGRAÇÃO COM RECURSOS/SERVIÇOS DE TERCEIROS

A partir desse documento, vimos que existem infinitas possibilidades de criações de programas, cada qual com diversos objetivos e meios de se fazer, visto que tudo depende da lógica do programador- sendo de bom tom nesse nicho profissional sempre colocar comentários percorrendo o código para facilitar a manutenção e aprimoramentos futuros.

Partindo desse pressuposto, na área da tecnologia sempre há modos de se automatizar cada vez mais, já que o ramo profissional continua a crescer e permitir mais conexões pelo globo. Seguindo esse raciocínio, existe também a possibilidade de conectar um programa a uma API (interface de de programação de aplicações).

API é um conjunto de padrões que participam de uma interface, a qual tem como intuito criar plataformas mais práticas aos desenvolvedores a partir da comunição entre dois sistemas. Desse modo, é responsável por ligar os recursos necessários para o software ter o desempenho desejável, sendo também usado para aprimiorar e otimizar tarefas/sistemas que já existem (integração de soluções de pagamento, lançamentos contábeis e emissão de notas fiscais, por exemplo). Contudo, pode-se dizer que as API´s contribuem para uma melhor performance, além de contar com poucos dados – só são inseridas as informações imprescindíveis ao objetivo- e com uma barreira de proteção específica para cada parte do sistema.

Em suma, as API´s podem se dividir em quatro categorias: privada – acesso exclusivo a empresa que criou-, composta – combinação de vários dados/serviços, nos quais os desenvolvedores podem acessar por meio de diferentes terminais-, pública – possui restrinção mínima, ou seja, um volume maior de pessoas tem acesso para usufruir da API- e parceira – as API´s são expostas estrategicamente aos parceiros empresarias a partir de um processo de integração -.

6.1. JSON E O RUBY.

Nesse projeto, decidimos utilizar uma API para integrar com o sistema de GPS do Google Maps. Para isso foi necessário instalar uma variente de extensão do JSON, porque é mais rápido do que a variante Ruby pura. Para tal ação, basta digitar na linha de comando como root:

```
# rake install_pure
```

O comando abaixo irá construir as extensões e instalá-las em seu sistema ou ainda pode optar por instalar apenas a implementação Ruby pura de JSON. No caso, se utiliza Rubygems, é necessário inserir a seguinte linha de comando:

```
# ruby install.rb
```

```
# gem install json
```

Agora, se optar utilizar a versão JSON mais recente, há também uma variante do Ruby Json Puro da Gema que pode ser instalada com:

```
# gem install json_pure
```

Logo, para utilizar o JSON pode optar por um dos comandos:

```
require 'json/pure'
```

```
require 'json'
```

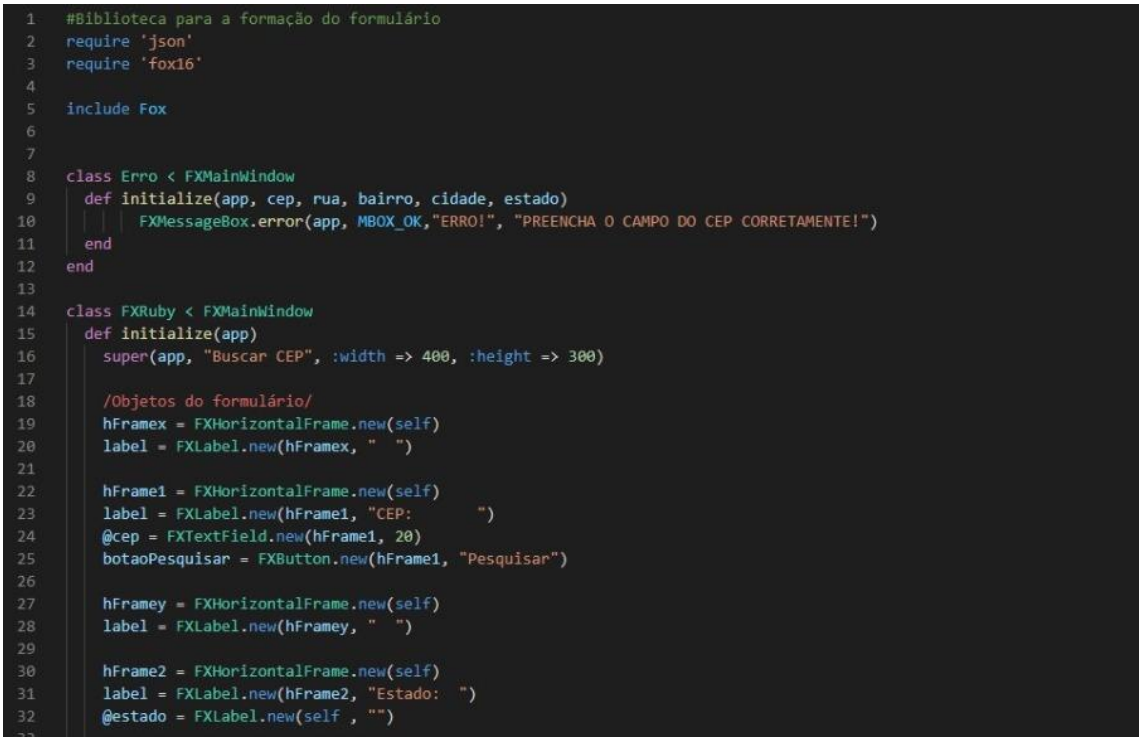
Por fim, para carregar a variante instalada (a extensão ‘json’ ou a variante pura ‘json_pura’):

```
require 'json/ext'
```

Sendo importante ressaltar que somente os ‘require’ são no escopo do código, os demais comandos são para serem realizados no prompt de comando.

6.2. UTLIZANDO JSON EM RUBY

Após a instalação, o grupo desenvolveu um programa com o intuito de utilizar o CEP, o qual seria indicado pelo usuário, e, por meio dele, descobrir a localização exata daquele número inserido, isso seria feito com base no sistema GPS do Google Maps. Verifique a seguir:



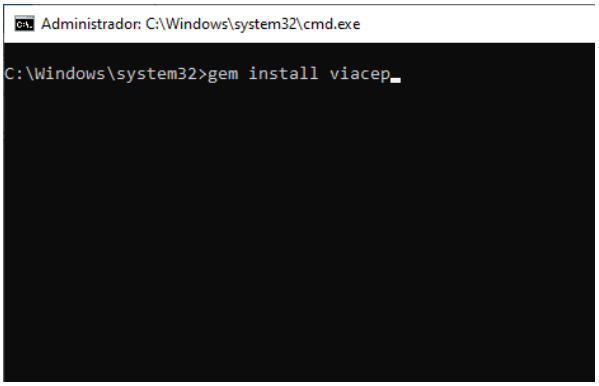
```ruby
#Biblioteca para a formação do formulário
require 'json'
require 'fox16'

include Fox


class Erro < FXMainWindow
  def initialize(app, cep, rua, bairro, cidade, estado)
      FXMessageBox.error(app, MBOX_OK,"ERRO!", "PREENCHA O CAMPO DO CEP CORRETAMENTE!")
  end
end

class FXRuby < FXMainWindow
  def initialize(app)
    super(app, "Buscar CEP", :width => 400, :height => 300)

    /Objetos do formulário/
    hFramex = FXHorizontalFrame.new(self)
    label = FXLabel.new(hFramex, "  ")

    hFrame1 = FXHorizontalFrame.new(self)
    label = FXLabel.new(hFrame1, "CEP:       ")
    @cep = FXTextField.new(hFrame1, 20)
    botaoPesquisar = FXButton.new(hFrame1, "Pesquisar")

    hFramey = FXHorizontalFrame.new(self)
    label = FXLabel.new(hFramey, "  ")

    hFrame2 = FXHorizontalFrame.new(self)
    label = FXLabel.new(hFrame2, "Estado:  ")
    @estado = FXLabel.new(self , "")
```

```
      hFrame3 = FXHorizontalFrame.new(self)
      label = FXLabel.new(hFrame3, "Cidade: ")
      @cidade = FXLabel.new(self , "")

      hFrame4 = FXHorizontalFrame.new(self)
      label = FXLabel.new(hFrame4, "Bairro:   ")
      @bairro = FXLabel.new(self , "")

      hFrame5 = FXHorizontalFrame.new(self)
      label = FXLabel.new(hFrame5, "Rua:      ")
      @rua = FXLabel.new(self , "")

      #Objeto de instância para utilizar na classe Erro
      botaoPesquisar.connect(SEL_COMMAND) do

        if @cep.text.size != 8
          obj = Erro.new(app, @cep, @rua, @bairro, @cidade, @estado)
        else
          obj = BuscaCEP.new(@app, @cep, @rua, @bairro, @cidade, @estado)

      end
    end
  end

#Classe para procurar CEP através de json e a API VIACEP
class BuscaCEP < FXMainWindow
  def initialize(app, cep, rua, bairro, cidade, estado)

    #Adicionando e instalando a gema, para utilização do VIACEP API
    link = "https://viacep.com.br/ws/"+ cep +"/json/?callback=?"

   #Localizando dados do CEP e preenchendo as labels
   cep.getJSON(link, function(dados) {
      #Atualiza os campos com os valores da consulta.
      rua.text = dados.logradouro;
      bairro.text = dados.bairro;
      cidade.text = dados.localidade;
      estado.text = dados.uf;
});
 end
end

def create
  super
    show (PLACEMENT_SCREEN)
  end
end

#Chamando o Form
app = FXApp.new
#Chamando a Classe
FXRuby.new(app)
#Criando o Form
app.create
#Mantendo o Form aberto
app.run
```

O programa acima deveria exibir um “Form” para a inserção de dados pelo usuário, método que já foi apresentado e explicado a partir do capítulo 3, e com o auxílio do JSON, a localização relacionada ao CEP digitado deveria aparecer, pois essa seria encontrada através da integração do sistema ao Google Maps. Entretanto, isso não funcionou. Conforme o desenvolver da aplicação, o grupo encontrou erros que não conseguiu resolver, e, dessa forma, esse método de utilização da API foi inviabilizado. Abaixo segue o erro que foi apresentado:

```
PROBLEMAS   SAÍDA   CONSOLE DE DEPURAÇÃO   TERMINAL

PS C:\Users\andre\OneDrive\Área de Trabalho\ruby> ruby cepErro.rb
cepErro.rb:69:in `+': no implicit conversion of Fox::FXTextField into String (TypeError)
        from cepErro.rb:69:in `initialize'
        from cepErro.rb:52:in `new'
        from cepErro.rb:52:in `block in initialize'
        from C:/Ruby30-x64/lib/ruby/gems/3.0.0/gems/fxruby-1.6.44-x64-mingw32/lib/fox16/responder2.rb:55:in `onHandleMsg'
        from cepErro.rb:100:in `run'
        from cepErro.rb:100:in `<main>'
PS C:\Users\andre\OneDrive\Área de Trabalho\ruby>
```

Após pesquisas, a viabilização da aplicação tornou-se possível por meio da utilização de uma “gema ruby”. As “gemas” no Ruby são como um grande conjunto de bibliotecas de terceiros, e, visto que não foi possível utilizar o JSON, decidiu-se que a utilização da gema “ViaCep” foi escolhida como a forma mais viável de desenvolver e compilar o programa. Porém, essa “gema” só pode ser chamada no programa caso ela seja instalada através do CMD como administrador. Para que seja instalado veja abaixo:



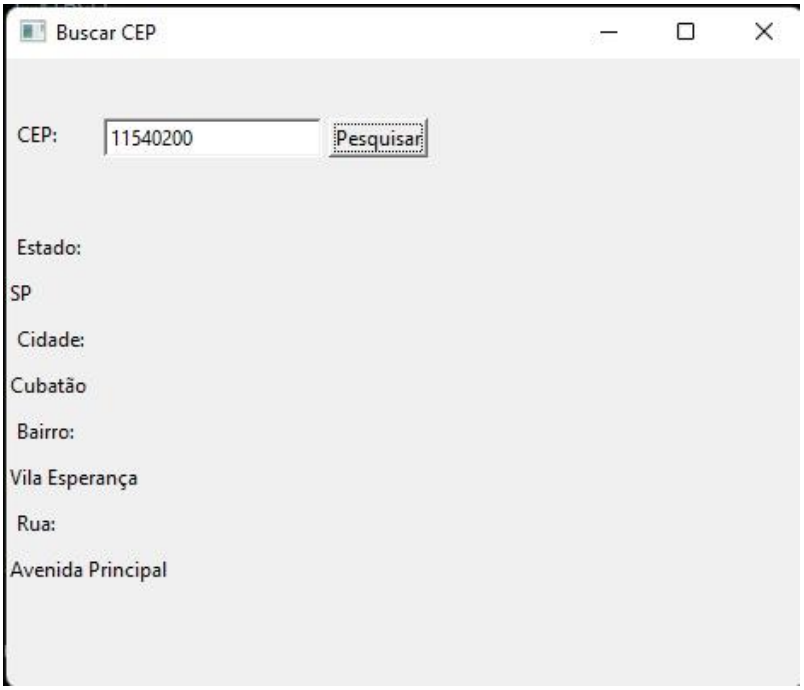
Com a instalação feita via CMD, o programa foi reescrito em função da nova forma de pesquisar o CEP, observe a seguir:

```ruby
#Classe para procurar CEP através da API VIACEP
class BuscaCEP < FXMainWindow
  def initialize(app, cep, rua, bairro, cidade, estado)
    /chamando a gema viacep/
    require 'viacep'

    auxCep = ViaCep::Address.new(cep)

    cidade.text = auxCep.city
    rua.text = auxCep.address
    bairro.text = auxCep.neighborhood
    estado.text = auxCep.state

  end
end

def create
  super
    show (PLACEMENT_SCREEN)
  end
end

#Chamando o Form
app = FXApp.new
#Chamando a Classe
FXRuby.new(app)
#Criando o Form
app.create
#Mantendo o Form aberto
app.run
```

Com o foco principal sendo a exibição do local através do uso do CEP, foi necessário implantar a classe erro, para que caso haja alguma informação errada ou incompleta, o usuário consiga corrigir sem receber alguma informação errada ou problema na compilação. Logo em seguida, foi preciso haver a criação de um botão para facilitar o entendimento do usuário, indicando onde ele deve clicar para receber suas informações. O botão pesquisar gera os resultados que são chamados cada um com sua variável correspondente (Estado, cidade, bairro e rua), gerando assim as informações devidas.

Percebe-se que a única mudança (entre o primeiro programa e o segundo) necessária para que o programa funcionasse foi a da classe BuscaCEP. Nela, antes era constado a busca pelo CEP por meio do JSON, mas agora ela é feita da maneira adequada para a funcionalidade da gema ViaCEP. Nessa classe, a gema é chamada por meio do “require” e relaciona os dados definidos pelo usuário com os objetos da gema, como nas linhas a seguir:

```
cidade.text = auxCep.city
rua.text = auxCep.address
bairro.text = auxCep.neighborhood
estado.text = auxCep.state
```



E, somente dessa forma, o grupo conseguiu compilar corretamente e fazer uso do programa.

REFERÊNCIAS

ADRIANO, Thiago S. **Ruby: O que é uma GEN**.Disponível em: <https://medium.com/rubybr/ruby-o-que-%C3%A9-uma-gem-d62aafc30724> Acesso em: 16 de Julho de 2021.

A linguagem de programação Ruby. Índice TIOBE. Disponível em:<https://www.tiobe.com/tiobe-index/ruby/>. Acesso em 17/05/2021

BARRETO, Marcelo. **Gem ViaCEP**. GITHUB. Disponível em: https://github.com/marcelobarreto/via_cep. Acesso em: 03 dez. 2021.

BRANDÃO, Paula. **VOCÊ sabe o que é API de integração? Entenda de uma vez por todas!.** Maplink, [*s. l.*], 13 jan. 2020. Disponível em: https://maplink.global/blog/o-que-e-api/. Acesso em: 29 nov. 2021.

Comunidade RUBY. **Bibliotecas em RUBY**. RUBY. Disponível em: https://www.ruby-lang.org/pt/libraries/. Acesso em: 28 nov. 2021.

DOMINIO PÚBLICO. **Consulte todos os CEPs do Brasil.** VIACEP. Disponível em: https://viacep.com.br/modulos_e_pacotes/. Acesso em: 28 nov. 2021.

ELIAS, Marcos. **O que é interface gráfica.** EXPLORANDO. Disponível em: < http://www.explorando.com.br/o-que-e-interfacegrafica/> Acesso em: 15 de Julho de 2021.

FABRO, Clara. **O que é API e para que serve? Cinco perguntas e respostas: Entenda o que são APIs e como são utilizadas por desenvolvedores**. **TechTudo**, [*s. l.*], 15 jun. 2020. Disponível em: https://www.techtudo.com.br/listas/2020/06/o-que-e-api-e-para-que-serve-cinco-perguntas-e-respostas.ghtml. Acesso em: 29 nov. 2021.

HIBBARD, James. **Uma introdução ao FXRuby**. SITEPOINT. Disponível em: <https://www.sitepoint.com/an-introduction-to-fxruby/> Acesso em: 7 de Julho de 2021.

ÍndiceTIOBE de maio de 2021. Índice TIOBE. Disponível em: <https://www.tiobe.com/tiobe-index/>. Acesso em 17/05/2021.

Interfaces GraficasconFXRuby - Llamado de Interfaces Externas, 2018. 1 vídeo (6 min 7 segs). Publicado pelo Programacion JJE. Disponível em: <https://www.youtube.com/watch?v=j9kTok243MM>. Acesso em: 23 Agosto de 2021.

JOHNSON, Lyle. **FXRuby: Create Lean and Mean GUIs with Ruby**. 2008. ed. [S.l.]: Editora **O′Reilly**, 2008. p. 102-116. <https://res.infoq.com/articles/johnson-fxruby-book-excerpt/en/resources/FXRubyWidgets.pdf> Acesso em: 9 de Julho de 2021.

MATSUMOTO, Yukihiro. **FXRuby: um olhar rápido**. RUBY LEARNING. Disponível em: <http://rubylearning.com/satishtalim/fxruby.html> Acesso em: 7 de Julho de 2021.

KENZIE. BANCO DE DADOS: O QUE É, PARA QUE SERVE, TIPOS E COMO **CRIAR.** Disponível em: <https://kenzie.com.br/blog/banco-de-dados/>. Acesso em: 11 out. 2021.

MISTRY, Jigar. **8 Best Ruby on Rails IDE and Text Editors for Web Development.** MONOCUBED. Disponível em: <https://www.monocubed.com/best-ruby-on-rails-ide/> Acesso em: 17 de maio de 2021.

NOLETO, Cairo. **Ruby: tudo sobre essa linguagem orientada a objetos!.**Blog Betrybe, [*s. l.*], 18 abr. 2020. Disponível em: https://blog.betrybe.com/linguagem-de-programacao/ruby/#:~:text=A%20linguagem%20Ruby%20foi%20criada,%2C%20Eiffel%2C%20Ada%20e%20Lisp. Acesso em: 16 maio 2021.

ORACLE .**O QUE É BANCO DE DADOS?**. Disponível em: <https://www.oracle.com/br/database/what-is-database/>. Acesso em: 11 out. 2021.

Quais são os tipos de APIs?. **LinkApi**, [*s. l.*], 6 jul. 2021. Disponível em: https://www.linkapi.solutions/blog/quais-sao-os-tipos-de-apis. Acesso em: 27 nov. 2021.

Ruby: O melhor amigo do programador. Ruby-Lang. Disponível em: https://www.ruby-lang.org/pt/about/. Acesso em: 16 maio 2021.

SILVA, Réulison. 10 linguagens de programação mais utilizadas no mercado.**DEVMEDIA,** 2019. Disponivel em: <https://www.devmedia.com.br/top-10-linguagens-de-programacao-mais-usadas-no-mercado/39635>. Acesso em: 18 de maio de 2021.

SQLITE.**SQLITE: ABOUT**. Disponível em: <https://www.sqlite.org/about.html> Acesso em: 07 out. 2021.

SOUZA, de Ivan. **O que é SQLite, por que ele é usado, e o que o diferencia do MySQL.** ROCKCONTENT.Disponível em: <https://rockcontent.com/br/blog/sqlite/> Acesso em: 07 out. 2021.

SOUZA, Lucas. **Ruby**: **Aprenda a programar na linguagem mais divertida**. 2014. ed. [S.l.]: Editora **Casa do Código**, 2014. p. 1-285. <https://books.google.com.br/books?hl=pt-BR&lr=lang_pt&id=qGSCCwAAQBAJ&oi=fnd&pg=PP6&dq=para+que+problema+o+ruby+foi+criado+&ots=OcjEMymYoP&sig=0rShGTi4GI_zO15evPY0Msjlk10#v=onepage&q&f=false> Acesso em: 20 de maio de 2021.

Vibryt. **O que são APIs e suas vantagens**. **Medium**, [*s. l.*], 4 fev. 2020. Disponível em: https://medium.com/peexell/o-que-s%C3%A3o-apis-e-suas-vantagens-7d3c68a068fa. Acesso em: 27 nov. 2021.

WALKER, Aaron. **What's the Best Ruby IDE for Web Development in 2019.**LEARN HUB. Disponível em: <https://learn.g2.com/ruby-ide> Acesso em: 17 de maio de 2021.