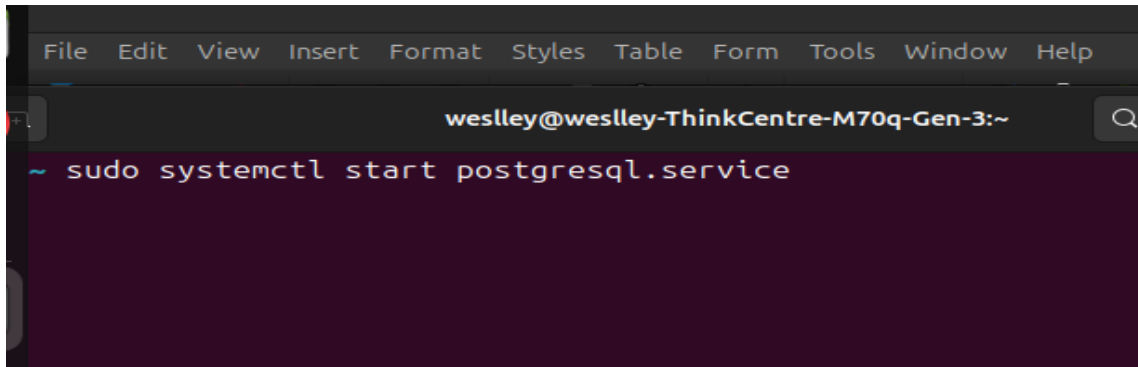


RUBY RAILS

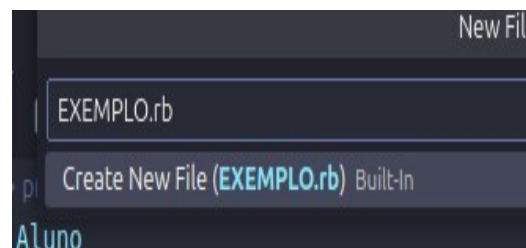
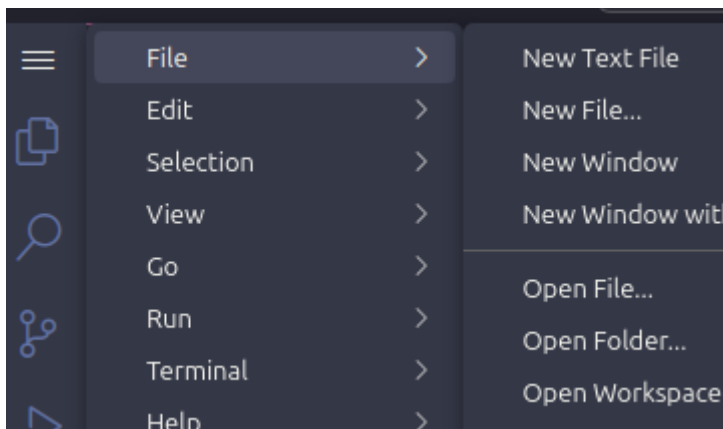
1.CRIAÇÃO E MANIPULAÇÃO DE ARQUIVOS RUBY

Para começar a manipulação de arquivos em Ruby, é necessário criar um arquivo do tipo Ruby (.rb). Você pode optar por criar uma aplicação inteira ou apenas um arquivo específico.

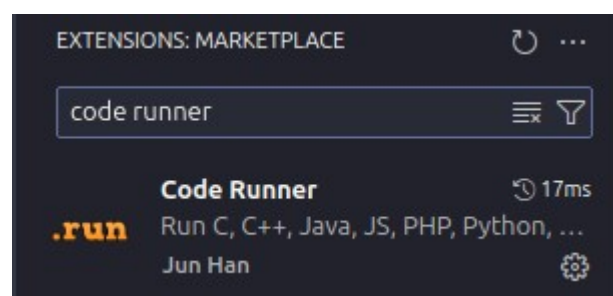
Caso opte por criar a aplicação, utilize o comando `sudo systemctl start postgresql.service` no terminal para iniciar o serviço necessário.



Se preferir criar apenas um arquivo, basta utilizar a opção "New File" e dar o nome desejado, adicionando a extensão `.rb` no final. Isso criará um arquivo Ruby, pronto para ser editado e manipulado.




Além disso, para executar a página e obter resultados ao utilizar o Ruby, é necessário ter um compilador e um editor de texto. Neste caso, estamos utilizando o Visual Studio Code (VS Code) como editor e a extensão Code Runner para facilitar a execução do código diretamente no ambiente de desenvolvimento.



2.INICIANDO RUBY


Ao iniciamos o estudo acerca de ruby vamos iniciar com alguns métodos básico:

a.PUTS: O comando `puts` é utilizado para imprimir informações ou resultados de variáveis e expressões no console. Depois de exibir o valor, ele adiciona uma quebra de linha automaticamente.

```
home > wesley > projetos > ruby >  hello.rb
1  puts "Olá, Mundo!" # Exibe "Olá, Mundo!" e pula para a próxima linha
2
```

```
[Running] ruby "/home/wesley/projetos/ruby/hello.rb"
Olá, Mundo!
```


b.PRINT: Similar ao `puts`, mas o `print` não adiciona automaticamente uma nova linha após a impressão do valor. Isso significa que, se você chamar `print` várias vezes, o texto será impresso na mesma linha.

```
home > wesley > projetos > 
1  print "hello"
2  print "mundo"

[Running] ruby "/home/wesley/projetos/ruby/hello.rb"
hellomundo
```


3.ASPAS: São utilizadas para definir padrões de comportamento da resposta dos métodos

a.Aspas simples ('): Quando você usa aspas simples, o conteúdo da string é interpretado exatamente como está, sem a possibilidade de interpolação (inserção de variáveis ou expressões) ou escape de caracteres especiais, como `\n` (quebra de linha), `\t` (tabulação).

```
e > wesley > projetos > ruby >  hello.rb
str = 'Olá, mundo!'
puts str # Exibe: Olá, mundo!
```

2. Aspas Duplas ("")

Quando você usa aspas duplas, Ruby **interpreta** o conteúdo da string e permite a **interpolação de variáveis** e a **interpretação de caracteres especiais** como `\n` (quebra de linha), `\t` (tabulação), entre outros.

```
home > wesley > projetos > ruby >  hello.rb
1  nome = "João"
2  puts "Olá, #{nome}!"
3  #exibir Olá, João!
```

3.3 Aspas(“”):Permite multilinhas sem usar caracteres especiais e a tabulação

```
home > wesley > projetos > ruby >  hello.rb
1  texto= """ viaosdad sadnabsd
2  saihdioashdiasd
3  osajdpoasjdo
4  """
5  puts texto
6  #Running] ruby "/home/wesley/projetos/ruby/hello.rb"
7  #viaosdad sadnabsd
8  #saihdioashdiasd
9  #osajdpoasjdo
```

3.VARIÁVEIS

a. Variáveis Locais

- **Definição:** São variáveis que são acessíveis apenas dentro do método ou bloco onde foram declaradas.
- **Sintaxe:** Começam com uma letra minúscula ou um sublinhado (_).

```
home > wesley > projetos > ruby >  hello.rb
1  nome= "weslley"
2  puts "hello #{nome}"
3  #[Running] ruby "/home/wesley/projetos/ruby/hello.rb"
4  #hello weslley
```

b.Variáveis de Instância

- **Definição:** São variáveis que são associadas a uma instância de uma classe e podem ser acessadas em qualquer método dentro dessa instância.
- **Sintaxe:** Começam com um @.

```
home > wesley > projetos > ruby >  hello.rb
1  class Pessoa
2  |  def initialize(nome)
3  |  |  @nome = nome # variável de instância
4  |  end
5  |
6  |  def saudacao
7  |  |  puts "Olá, #{@nome}!" # acessando a variável de instância
8  |  end
9  end
10
11  pessoa = Pessoa.new("Weslley")
12  pessoa.saudacao # Output: Olá, Weslley!
```

1.O método `initialize` é um **método especial** chamado automaticamente quando um novo objeto da classe é criado.

2.Define um método `saudacao`, que imprime uma mensagem personalizada com o nome da pessoa.

3.Cria um novo objeto da classe `Pessoa`, passando "Maria" como argumento. O método `initialize` é chamado, e "Maria" é armazenado na variável de instância `@nome`.

lass Aluno: Define uma classe chamada `Aluno`.

```
home > wesley > projetos > ruby > hello.rb
1  class Aluno
2    attr_accessor :nome, :idade, :cidade # cria automaticamente os métodos getter e setter para os atributos da classe.
3
4    def initialize(nome, idade, cidade)
5      @nome = nome
6      @idade = idade
7      @cidade = cidade
8    end
9  end
10
11  aluno = Aluno.new("Wesley", 27, "Porto Velho")
12  puts "#{aluno.nome}, #{aluno.idade} anos, mora em #{aluno.cidade}."
13
```

- Linha 1:Uma **classe** é como um molde para criar objetos.
- Linha2:`attr_accessor` cria automaticamente **métodos de leitura e escrita** para os atributos **nome**, **idade** e **cidade**. Isso significa que podemos **ler e modificar** esses atributos diretamente.
- Linha4:O método **`initialize`** é chamado automaticamente quando criamos um novo aluno. Ele recebe três **parâmetros**: `nome`, `idade` e `cidade`.
- **Linha 5-7:Os valores são armazenados nas variáveis de instância** `@nome`, `@idade` e `@cidade`, para que possam ser usados depois.
- Linha 11:Isso cria um novo objeto `aluno1` e atribui:
 - `@nome = "Maria"`
 - `@idade = 25`
 - `@cidade = "Rio de Janeiro"`
- **Interpolação de Strings** (`#{}`) é usada para exibir os valores armazenados nos atributos do objeto.
- Criamos um novo objeto da classe `Aluno`.
- Carlos, 30 anos, mora em São Paulo.