



**Universidade de Brasília**

**SHINY**

.....

**INTERATIVIDADE COM R**

**Davi, Eduardo, Gabriela, Jadson, Tailine**

# SUMÁRIO

1. O que é o Shiny?
2. Estrutura básica
  - 2.1 User Interface
  - 2.2 Server
3. Aplicações
4. Referências

*Shiny*

A white decorative swoosh underline that starts from the bottom of the letter 'y' and curves downwards and to the left, ending in a pointed, tail-like shape.

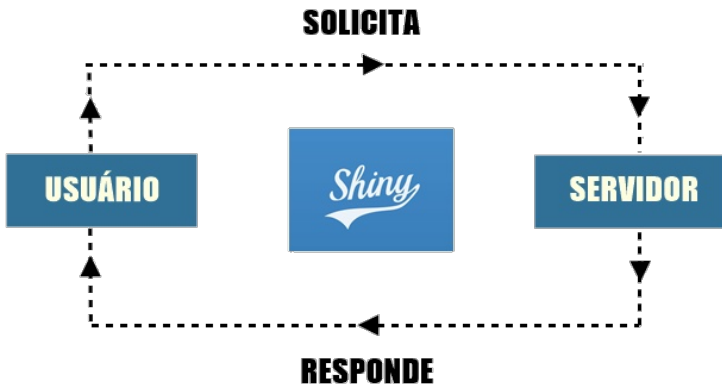
## INTERATIVIDADE

**Instruções do Servidor (R)**



**User Interface (UI)**

## REATIVIDADE



## RESUMINDO

Pacote do R

Cria de um servidor que envia páginas web, **recebe** informações do usuário e **processa** os dados.

Permite estruturar a interface do usuário **e** o processamento de dados.

Vantagens para o programador e para o usuário.

## ESTRUTURA

```
library(shiny)

ui <- fluidPage()

server <- function(input, output) {}

shinyApp(ui = ui, server = server)
```

## USER INTERFACE (UI)

Função	Finalidade
<code>library(shiny)</code>	Carregar o pacote Shiny.
<code>ui &lt;- fluidPage()</code>	Criar uma interface com o usuário.
<code>titlePanel()</code>	Criar um painel contendo um título do aplicativo.
<code>sidebarLayout()</code>	Criar um layout com uma barra lateral e área principal.
<code>sidebarPanel()</code>	Criar um painel com barra lateral.
<code>mainPanel()</code>	Criar um painel principal contendo elementos de saída.



# WIDGETS

## Buttons


`actionButton()`  
`submitButton()`

## Single checkbox

☒ Choice A

`checkboxInput()`

## Checkbox group

☒ Choice 1  
☐ Choice 2  
☐ Choice 3

`checkboxGroupInput()` `dateInput()`

## Date input

## Date range

 to 

`dateRangeInput()`

## File input

 No file chosen

`fileInput()`

## Numeric input

`numericInput()`

## Password Input

`passwordInput()`

## Radio buttons

☒ Choice 1  
☐ Choice 2  
☐ Choice 3

`radioButtons()`

## Select box

`selectInput()`

## Sliders


`sliderInput()`

## Text input

`textInput()`

© CC 2015 RStudio, Inc.

## CRIANDO FUNÇÕES DE ENTRADA

```
library(shiny)

ui <- fluidPage(
  sliderInput(inputId = "num",
              label = NULL,
              value = 25, min = 1, max = 100) )

server <- function(input, output) {}

shinyApp(ui = ui, server = server)
```

## CRIANDO FUNÇÕES DE ENTRADA

```
library(shiny)

ui <- fluidPage(
  sliderInput(inputId = "num",
              label = NULL,
              value = 25, min = 1, max = 100) )

server <- function(input, output) {}

shinyApp(ui = ui, server = server)
```

## PRÓXIMO PASSO

Para que seja possível **visualizar** o input, é necessário escolher como será o **output**. Para esse exemplo, queremos que o output gere um **gráfico**. Mas que **função** precisamos usar agora?

## OUTPUTS

Função	Finalidade
<code>dataTableOutput()</code>	Tabela Interativa
<code>htmlOutput()</code>	HTML puro
<code>imageOutput()</code>	Imagem
<code>plotOutput()</code>	Gráfico
<code>tableOutput()</code>	Tabela
<code>textOutput()</code>	Texto
<code>uiOutput()</code>	Elemento do Shiny UI
<code>verbatimTextOutput()</code>	Texto

## DEFININDO O TIPO DE OUTPUT

```
library(shiny)
ui <- fluidPage(
  sliderInput(inputId = "num",
              label = NULL,
              value = 25, min = 1, max = 100),
  plotOutput("hist"))
server <- function(input, output) {}
shinyApp(ui = ui, server = server)
```

## DEFININDO O TIPO DE OUTPUT

```
library(shiny)

ui <- fluidPage(

  sliderInput(inputId = "num",

              label = NULL,

              value = 25, min = 1, max = 100),

  plotOutput("hist"))

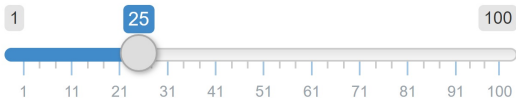
server <- function(input, output) {}

shinyApp(ui = ui, server = server)
```

## RESULTADO

Agora foi gerado um **botão de slide** onde o usuário fará a escolha de um número entre 1 e 100.

**Choose a number**





## PRÓXIMO PASSO

A próxima etapa é **configurar** o output.

Dentro do UI, apenas demos alguns nomes.

Agora precisamos definir o que realmente vai acontecer.

## SERVER

Função	Finalidade
<code>library(shiny)</code>	Carregar o pacote Shiny.
<code>shinyServer()</code>	Definir a lógica do servidor do aplicativo Shiny.
<code>function(input,output)</code>	Funções <code>render()</code>

## RENDER ()

Output (UI)	Render (Server)
dataTableOutput()	renderDataTable
imageOutput()	renderImage
plotOutput()	renderPlot
tableOutput()	renderTable
textOutput()	renderText
verbatimTextOutput()	renderPrint
uiOutput()	renderUI
htmlOutput()	renderUI

## CONFIGURANDO O OUTPUT

```
library(shiny)

ui <- fluidPage(

  sliderInput(inputId = "num",

              label = NULL,

              value = 25, min = 1, max = 100),

  plotOutput("hist"))

server <- function(input, output) {

  output$hist <- renderPlot({

    hist(rnorm(input$num))})}

shinyApp(ui = ui, server = server)
```

## CONFIGURANDO O OUTPUT

```
library(shiny)

ui <- fluidPage(

  sliderInput(inputId = "num",

              label = NULL,

              value = 25, min = 1, max = 100),

  plotOutput("hist"))

server <- function(input, output) {

  output$hist <- renderPlot({

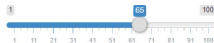
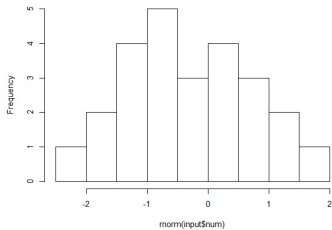
    hist(rnorm(input$num))})}

shinyApp(ui = ui, server = server)
```

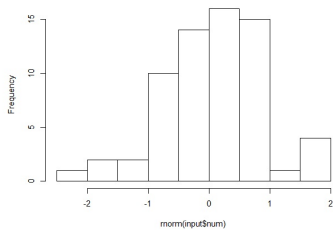
# RESULTADO



Histogram of `rnorm(input$num)`



Histogram of `rnorm(input$num)`



# APLICAÇÕES

## REFERÊNCIAS

1. RSTUDIO INC. **Shiny from RStudio**. Disponível em:  
<https://shiny.rstudio.com/tutorial/>. Acesso em: setembro de 2019.
2. PUC MINAS. **Desenvolvimento de Aplicativos Web Com R e Shiny**: inovações no ensino de Estatística. Belo Horizonte,v. 6, n. 2, p. 55-71, maio 2018
3. **Curso-R**. Disponível em: <http://material.curso-r.com/shiny/>. Acesso em: setembro de 2019.