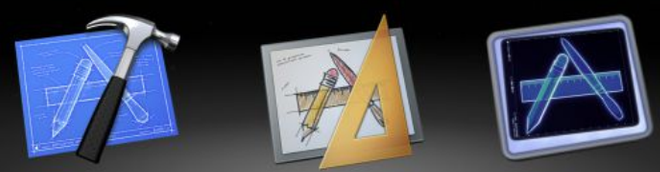
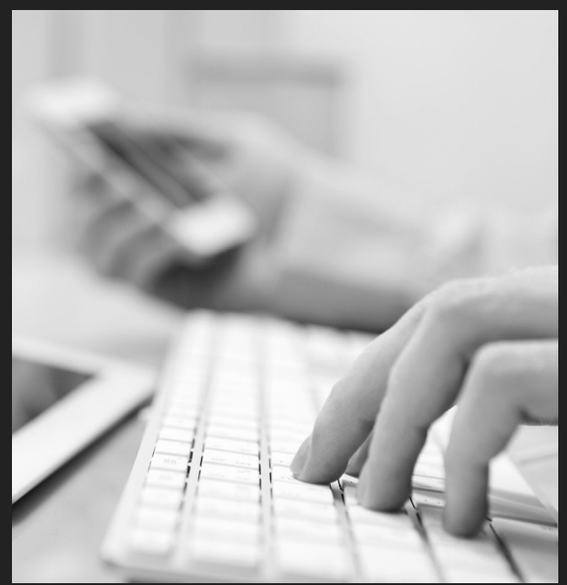


PROF. RENÊ XAVIER

DESENVOLVIMENTO PARA IOS 11 COM SWIFT 4

AGENDA

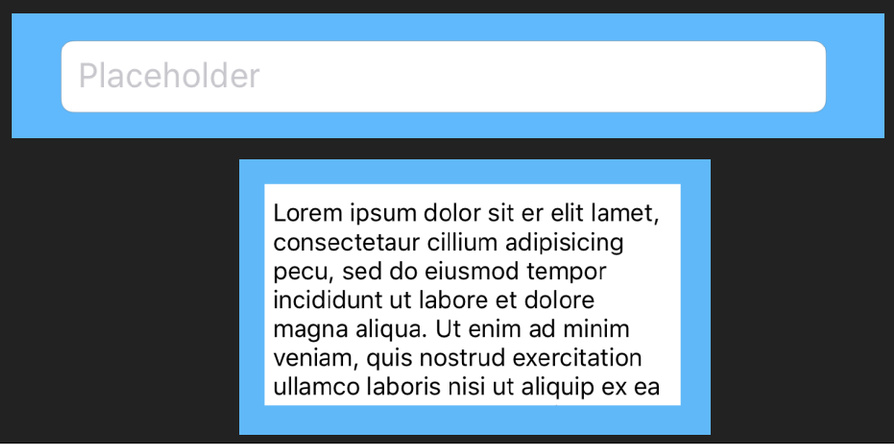
- Recebendo Input do usuário
- Navegação - Passando Informações entre telas



RECEBENDO INPUT DO USUÁRIO

RECEBENDO INPUT DO USUÁRIO

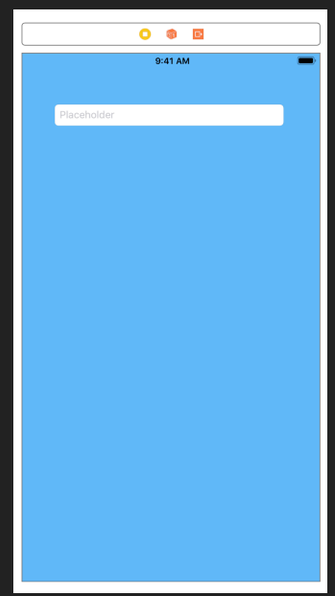
- Duas formas de inserir texto:
  - Text Field
  - Text View



RECEBENDO INPUT DO USUÁRIO

- Crie uma ViewController com um Text Field
- Crie uma classe
- Faça a conexão do TextField com o código

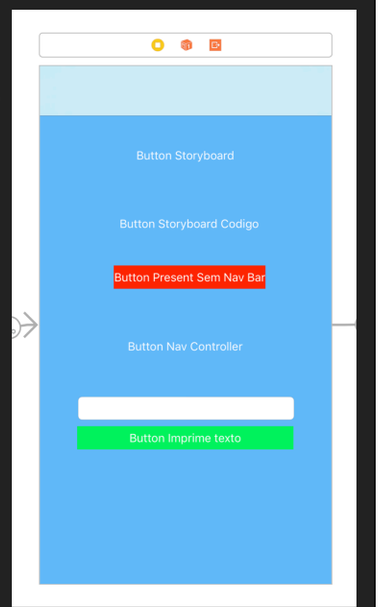
```
@IBOutlet weak var textoUsuario: UITextField!
```



RECEBENDO INPUT DO USUÁRIO

- Crie um botão
- Crie uma IBAction, ligando ele ao código quando clicado
- Faça ele imprimir o que está no UITextField

```
@IBAction func escreverTextoDigitado(_ sender: Any) {
    print(textoUsuario.text ?? "nil")
}
```



Explicar Debugger

RECEBENDO INPUT DO USUÁRIO

POR QUE DO .TEXT? ATÉ UILABEL TEM

```
@IBAction func escreverTextoDigitado(_ sender: Any) {
    print(textoUsuario.text ?? "nil")
}
```

Pois um uitextfield, Ullabel não são compostos apenas de texto. O texto é só uma property dele.

PASSANDO INFORMAÇÕES ENTRE TELAS

PASSANDO INFORMAÇÕES ENTRE TELAS

- Para a próxima tela (não conectado por Segue - Programático)
- Para a próxima tela (com Segue - Storyboard)

MAIS UTILIZADOS

- Para a tela anterior (Segue Unwind)

POUQUISSIMO UTILIZADO

- Utilizando o Padrão de Projeto Delegate
- Utilizando uma Closure ou Completion Handler

IMPORTANTÍSSIMO DOMINAR

- Utilizando o Padrão de Projeto Observer

UTILIZE COM CUIDADO

PASSANDO INFORMAÇÕES ENTRE TELAS

O QUE NÃO FAZER?

- Não o UserDefaults como armazenamento de variáveis de telas

PASSANDO INFORMAÇÕES ENTRE TELAS

O QUE NÃO FAZER?

- Não utilize um Singleton para passar dados de uma tela para a outra

```
class PassarDados {
    static var shared = PassarDados()
    private init() {}

    var texto: String?
}
```

```
PassarDados.shared.texto = "Texto a ser enviado"
```

Usar singleton para Logs ou Threads compartilhando um recurso único

PASSANDO INFORMAÇÕES ENTRE TELAS

O QUE NÃO FAZER?

- Não utilize o AppDelegate
  - É como ter um Singleton

```
let appDelegate = UIApplication.shared.delegate as! AppDelegate
let variavel = appDelegate.variavel
```

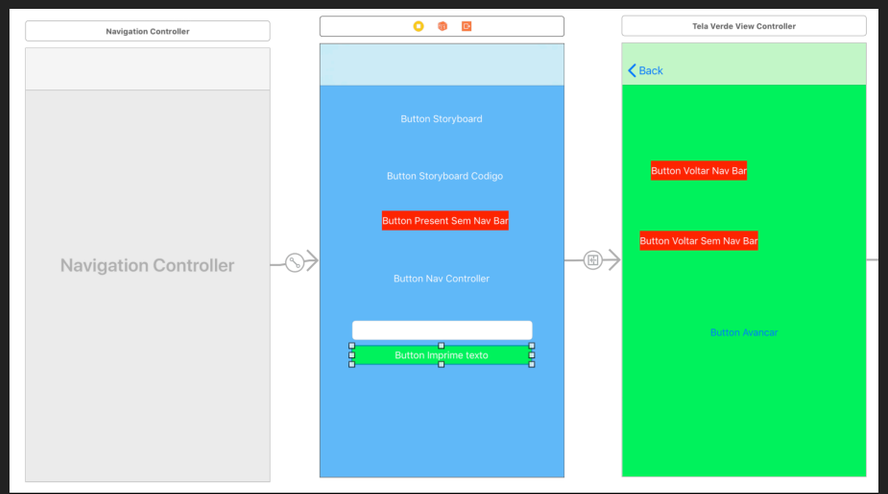
PASSANDO INFORMAÇÕES ENTRE TELAS

PARA QUE SERVE O APP DELEGATE MESMO?

- Configurações iniciais do App - applicationDidFinishLaunching
- Verificar quando fazer uma migração do banco de dados
- Configurações de bibliotecas (quando elas pedem)
- "Limpeza" do app ao sair - applicationWillTerminate

PARA PRÓXIMA TELA SEM SEGUE

PARA PRÓXIMA TELA SEM SEGUE




```
let storyboardMain = UIStoryboard(name: "Main", bundle: nil)
let telaVerdeViewController = storyboardMain.instantiateViewController(withIdentifier: "telaVerdeStoryboardID")
self.navigationController?.pushViewController(telaVerdeViewController, animated: true)
```

PARA PRÓXIMA TELA SEM SEGUE

TELA DESTINO

- Criamos a property que vamos receber o valor

```
class TelaVerdeViewController: UIViewController {
    var texto: String?
```

TIPO STRING?

Sem interrogação, ele não pode ser nulo então espera um valor

PARA PRÓXIMA TELA SEM SEGUE

TELA ORIGEM

- Definimos o tipo da ViewController
- Preenchemos a property

```
@IBAction func buttonNavControllerPressed(_ sender: Any) {
    let storyboardMain = UIStoryboard(name: "Main", bundle: nil)
    if let telaVerdeViewController = storyboardMain.instantiateViewController(withIdentifier:
        "telaVerdeStoryboardID") as? TelaVerdeViewController {
        telaVerdeViewController.texto = "oi"
        self.navigationController?.pushViewController(telaVerdeViewController, animated: true)
    }
}
```

PARA PRÓXIMA TELA SEM SEGUE

TELA ORIGEM

- Testar passando o texto que o usuário digitou

```
telaVerdeViewController.texto = textoUsuario.text
```

COMO VISUALIZAR SE DEU CERTO?

DEBUG

UM PAUSE PARA DEBUG

DEBUG

- Em qualquer classe que você clicar em cima do número da linha

```
19      override func viewDidLoad() {
20          super.viewDidLoad()
21      }
```

DEBUG

- Em qualquer classe que você clicar em cima do número da linha
- Uma indicação de debug vai aparecer

```
19      override func viewDidLoad() {
20          super.viewDidLoad()
21      }
```

Breakpoint condicional

DEBUG

- Em qualquer classe que você clicar em cima do número da linha
- Uma indicação de debug vai aparecer
- Se você clicar novamente, você desabilita

```
19      override func viewDidLoad() {
20          super.viewDidLoad()
21      }
```

DEBUG

- Em qualquer classe que você clicar em cima do número da linha
- Uma indicação de debug vai aparecer
- Se você clicar novamente, você desabilita
- Para remover você clica e arrasta em direção ao código, depois solta

DEBUG

- Em qualquer classe que você clicar em cima do número da linha
- Uma indicação de debug vai aparecer
- Se você clicar novamente, você desabilita
- Para remover você clica e arrasta em direção ao código, depois solta
- Antes do código executar aquela linha, ele para a execução

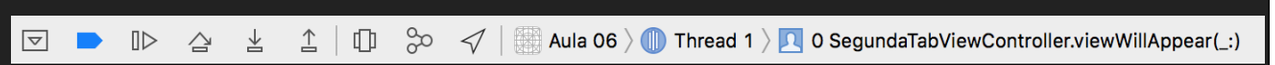
```
26      override func viewWillAppear(_ animated: Bool) {
27          super.viewWillAppear(animated)          Thread 1: breakpoint 8.1
```

DEBUG

- A tela de debug fica localizada na parte inferior do Xcode
- Pode ser exibida/escondida com Cmd + Shift + Y

DEBUG

- Esses são os controles de execução em Debug:
  - Esconde exibe a área de debug
  - Ativa/Desativa os breakpoints
  - Continua a execução após uma parada
  - Pula para a próxima linha de execução
  - Entra no método a ser executado



DEBUG

- Esses são os controles de execução em Debug:
  - Sai do método, mas para no método que chamou
  - Debug Visual
  - Debug de Memória
  - Simular Localização
  - Seleção da Thread

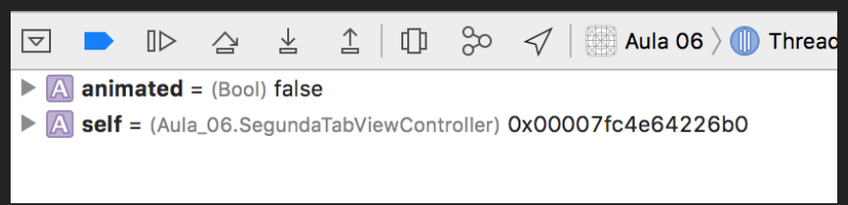
DEBUG

- No canto inferior direito você verá isso:

  - Selecione os dois

DEBUG

- No lado esquerdo:
  - Referência self
  - Variáveis da classe
  - Variáveis recebidas no método



DEBUG

- No lado direito:
  - Logs do sistema
  - Logs de biblioteca
  - Seus logs

```
c[60984]: Class VCWeakObjectHolder is implemented in both /Applications/Xcode.app/Contents/Developer
rtforms/iPhoneOS.platform/Developer/Library/CoreSimulator/Profiles/Runtimes/iOS.simruntime/Contents/
ources/RuntimeRoot/System/Library/PrivateFrameworks/AVConference.framework/Frameworks/
eroyTrace.framework/ViceroyTrace (0x126e824d0) and /Applications/Xcode.app/Contents/Developer/
rtforms/iPhoneOS.platform/Developer/Library/CoreSimulator/Profiles/Runtimes/iOS.simruntime/Contents/
ources/RuntimeRoot/System/Library/PrivateFrameworks/AVConference.framework/AVConference (0x125faee3
 of the two will be used. Which one is undefined.
wDidLoad oi
wWillAppear oi
wWillAppear oi
.db)
```

DEBUG

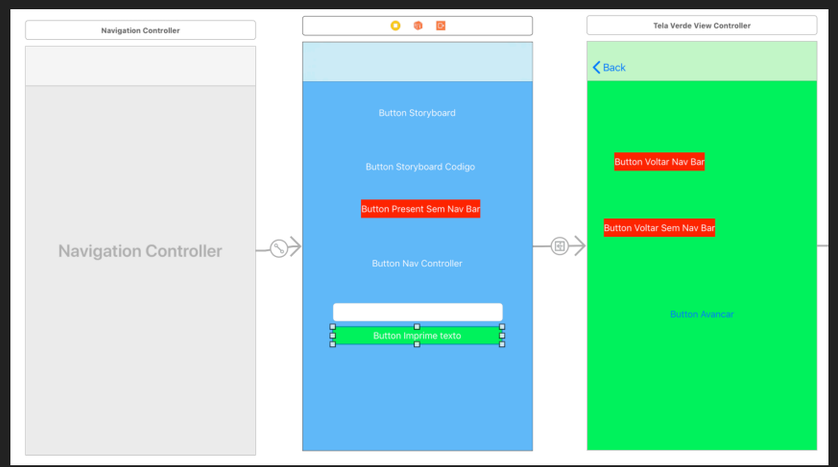
- No lado direito:
  - Logs do sistema
  - Logs de biblioteca
  - Seus logs
  - Você pode verificar e até mesmo alterar o valor de uma variável utilizando o comando:

```
po nomedavariavel

po nomedavariavel = valor
```

PARA PRÓXIMA TELA COM SEGUE

PARA PRÓXIMA TELA COM SEGUE



PARA PRÓXIMA TELA COM SEGUE

TELA DESTINO

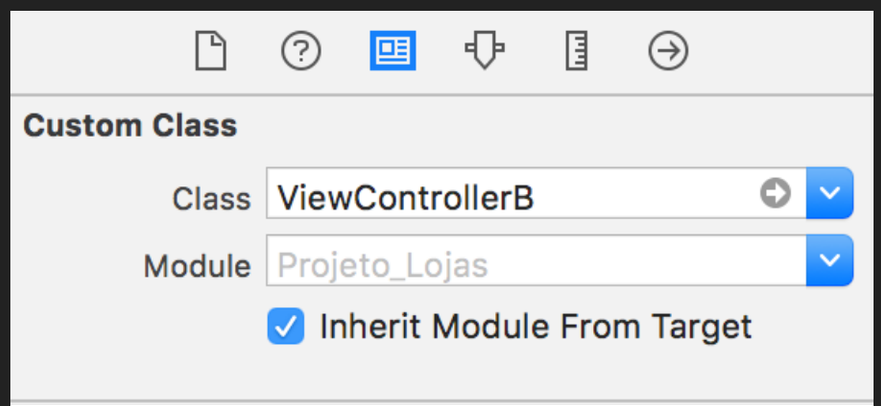
- Semelhante a anterior:
- Criar uma classe
- Na classe criar uma property

TELA ORIGEM

- Criar um Segue Identifier no Storyboard
- No método prepareForSegue determinar se é a segue específica
- Fazer como no anterior e preencher a property

PARA PRÓXIMA TELA COM SEGUE

TELA DESTINO



PARA PRÓXIMA TELA COM SEGUE

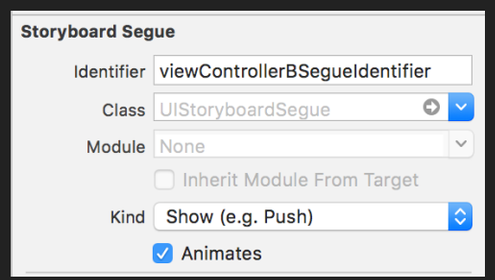
TELA DESTINO

```
11  class ViewControllerB: UIViewController {
12
13      var texto: String?
14
15      override func viewDidLoad() {
```

Pode ser interpretado como uma variável de classe

PARA PRÓXIMA TELA COM SEGUE

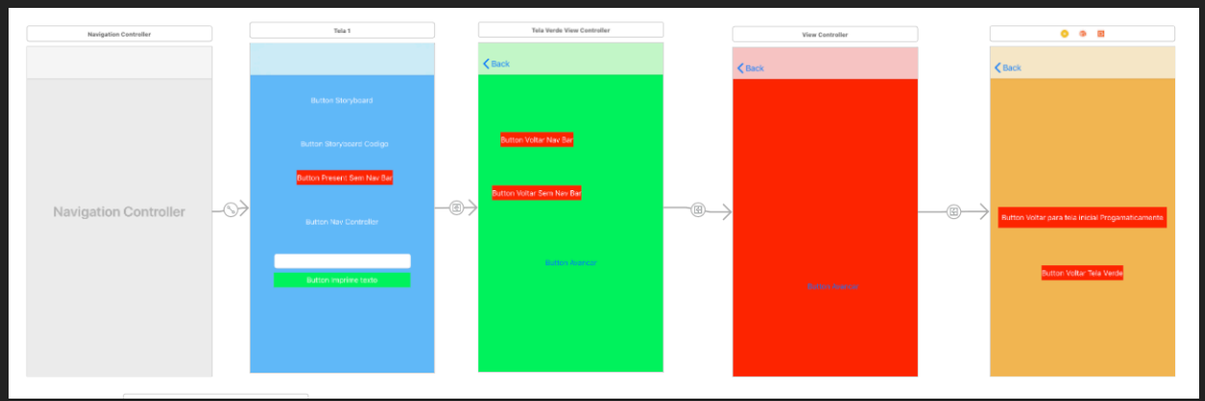
TELA ORIGEM – SEGUE STORYBOARD




```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if segue.identifier == "viewControllerBSegueIdentifier", let viewControllerB = segue.destination
        as? ViewControllerB  {
        viewControllerB.texto = textoUsuario.text
    }
}
```

PARA TELA ANTERIOR COM SEGUE UNWIND

PARA TELA ANTERIOR SEGUE UNWIND



Laranja para verde

PARA TELA ANTERIOR SEGUE UNWIND

TELA ORIGEM (ATUAL)

- Criamos uma property
- No prepareForSegue temos de preencher a property com o valor mais atual (caso já não o tenhamos feito)

TELA DESTINO (ANTERIOR)

- No método criado para fazer o unwind (que não foi conectado ao Storyboard) recuperamos o parâmetro que vem pela função que é UIViewController atual fazemos o cast e pegamos o valor da property.

PARA TELA ANTERIOR SEGUE UNWIND

TELA ORIGEM (ATUAL)

```
11  class TelaLaranjaViewController: UIViewController {
12      var unwind = "unwind var"
13
```

PARA TELA ANTERIOR SEGUE UNWIND

TELA DESTINO (ANTERIOR)

```
    @IBAction func voltarParaTelaVerde(_ sender: UIStoryboardSegue)  {
34      if let origin = sender.source as? TelaLaranjaViewController {
35          print(origin.unwind)
36      }
37  }
```

PARA TELA ANTERIOR COM DELEGATE

PARA TELA ANTERIOR DELEGATE

QUANDO USAR?

- Quando os outros casos não nos satisfazem
- Quando queremos passar um dado ao executar uma ação ou algo diferente de uma transição

PORQUE É IMPORTANTÍSSIMO?

- Esse padrão não é utilizado somente para passar informações para telas anteriores como criação de componentes.

PARA TELA ANTERIOR SEGUE UNWIND

TELA ORIGEM (ATUAL)

- Criamos um protocolo
- Criamos uma referência para uma classe que implementa esse protocolo
- Chamamos o método da referência que temos, então ele irá executar na referência

TELA DESTINO (ANTERIOR)

- Informamos que nossa classe implementa o protocolo da outra classe
- Implementamos os métodos do protocolo

Protocol é como uma interface do Java

PARA TELA ANTERIOR SEGUE UNWIND

TELA ORIGEM (ATUAL)

```
11  protocol TelaVerdeViewControllerDelegate: AnyObject {
12      func voltarPassandoArgumentos(_ texto:String)
13  }
14
15  class TelaVerdeViewController: UIViewController {
16      weak var delegate: TelaVerdeViewControllerDelegate?
```

Retain cycle

PARA TELA ANTERIOR SEGUE UNWIND

TELA ORIGEM (ATUAL)

- No momento que queremos podemos chamar o método do delegate.

```
    @IBAction func buttonVoltarPressed(_ sender: Any) {
        delegate?.voltarPassandoArgumentos("texto de retorno")
//          quando se utiliza a nav bar, para voltar a tela, usa o pop
        self.navigationController?.popViewController(animated: true)
    }
```

Retain cycle

PARA TELA ANTERIOR SEGUE UNWIND

TELA DESTINO (ANTERIOR)

```
11  class TelaInicialViewController: UIViewController, TelaVerdeViewControllerDelegate {
12      func voltarPassandoArgumentos(_ texto: String) {
13          print(texto)
14      }
```

DICA

PARA TELA ANTERIOR SEGUE UNWIND

TELA DESTINO (ANTERIOR) – EXTENSION

- Podemos deixar mais elegante, se no lugar do que fizemos na tela de destino fizermos o seguinte (remova o que você fez, se for fazer assim):
- No mesmo arquivo TelaInicialViewController.swift, na última línha fora da classe, coloque essa Extension:

```
69  extension TelaInicialViewController: TelaVerdeViewControllerDelegate {
70      func voltarPassandoArgumentos(_ texto: String) {
71          print(texto)
72      }
73  }
```

PARA TELA ANTERIOR SEGUE UNWIND

TELA DESTINO (ANTERIOR) – EXTENSION

- Uma Extension serve para "adicionar" novos métodos a classes já existentes (como a String, UIColor...).
- Nada impede a criação de uma Extension da sua própria classe que indique qual protocolo ela implementa.
- Facilita encontrar quais métodos estão relacionados com

```
69  extension TelaInicialViewController: TelaVerdeViewControllerDelegate {
70      func voltarPassandoArgumentos(_ texto: String) {
71          print(texto)
72      }
73  }
```

PARA TELA ANTERIOR SEGUE UNWIND

TELA DESTINO (ANTERIOR) – EXTENSION

- Extensions não afetam a alocação de memória.
- Na hora de gerar o app o Xcode só reúne esses códigos.

PARA TELA ANTERIOR SEGUE UNWIND

TELA DESTINO (ANTERIOR) – EXTENSION

- Uma Extension serve para "adicionar" novos métodos a classes já existentes (como a String, UIColor...).
- Nada impede a criação de uma Extension da sua própria classe que indique qual protocolo ela implementa.
- Facilita encontrar quais métodos estão relacionados com

```
69  extension TelaInicialViewController: TelaVerdeViewControllerDelegate {
70      func voltarPassandoArgumentos(_ texto: String) {
71          print(texto)
72      }
73  }
```