

UIKit e Alertas

Conhecendo outros componentes

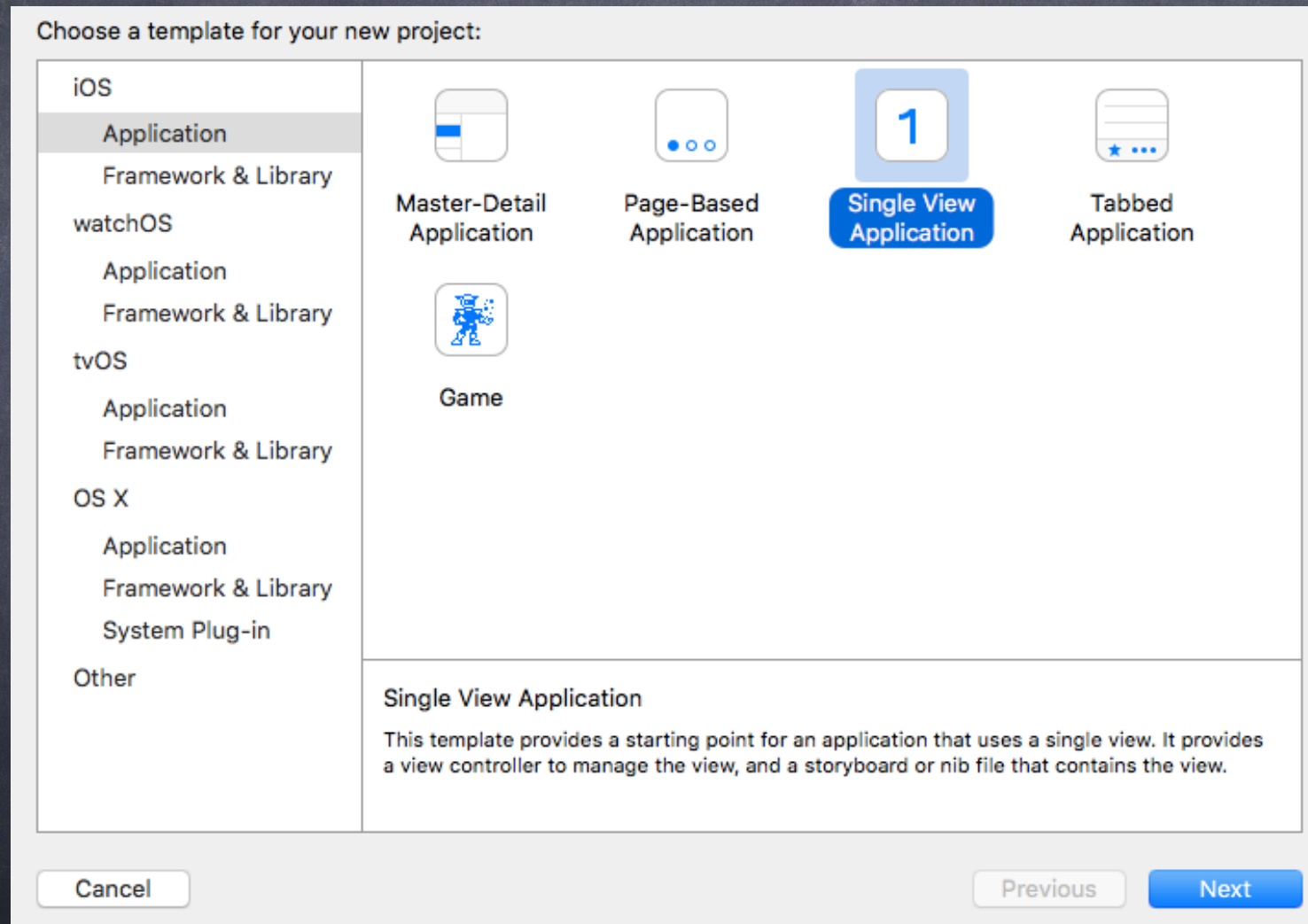
X-Code com Swift
Prof. Agesandro Scarpioni

Outros Componentes – UIKit

- Vamos criar um aplicativo para testarmos vários componentes que serão de grande utilidade no desenvolvimento de seus APP's, além de exibirmos mensagens para os usuários.

Iniciando o Projeto

- Clique em File -> New Project -> iOS -> Application -> Single View Application -> Next.



O App

- Preencha com os dados abaixo, lembre-se que o Organization Identifier é como se fosse o pacote no Java ou o namespace do VB, desmarque o 2º e 3º check box. Em Devices selecione iPhone, em Language escolha Swift.

Choose options for your new project:

Product Name:

Organization Name:

Organization Identifier:

Bundle Identifier:

Language:

Devices:

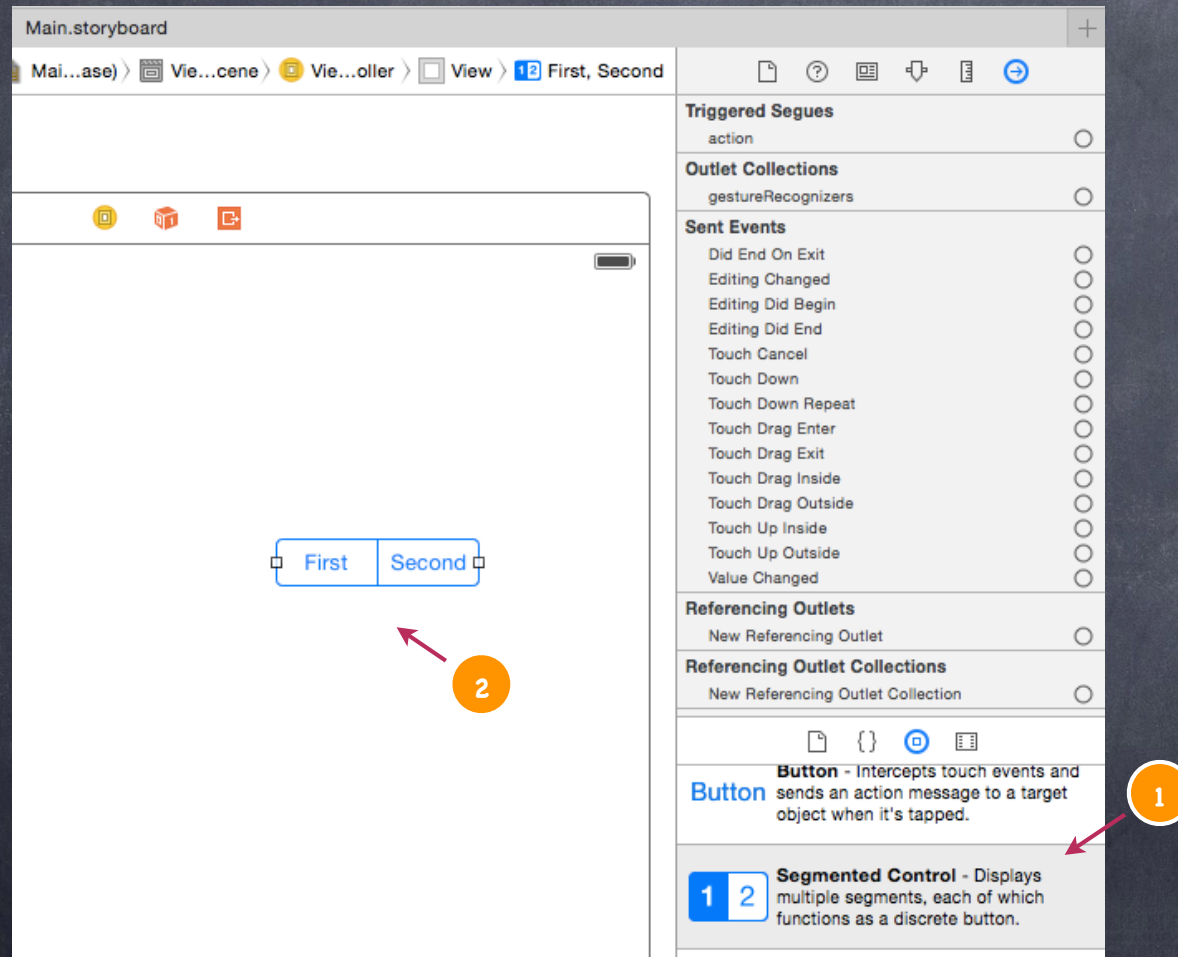
☐ Use Core Data

☐ Include Unit Tests

☐ Include UI Tests

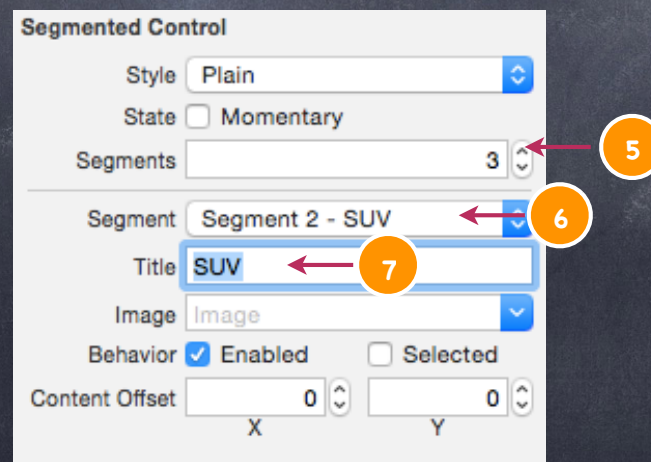
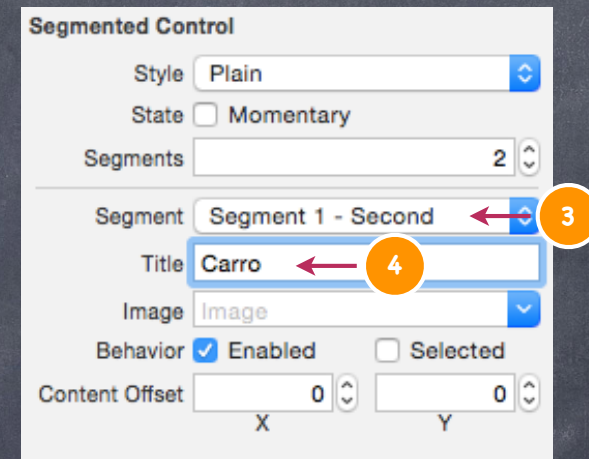
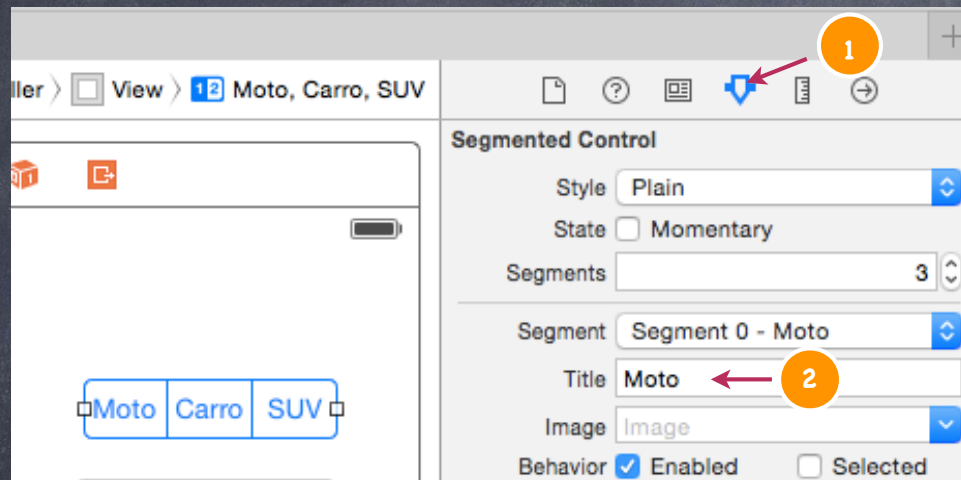
Segmented Control

- Coloque em sua tela um objeto do tipo Segmented Control.



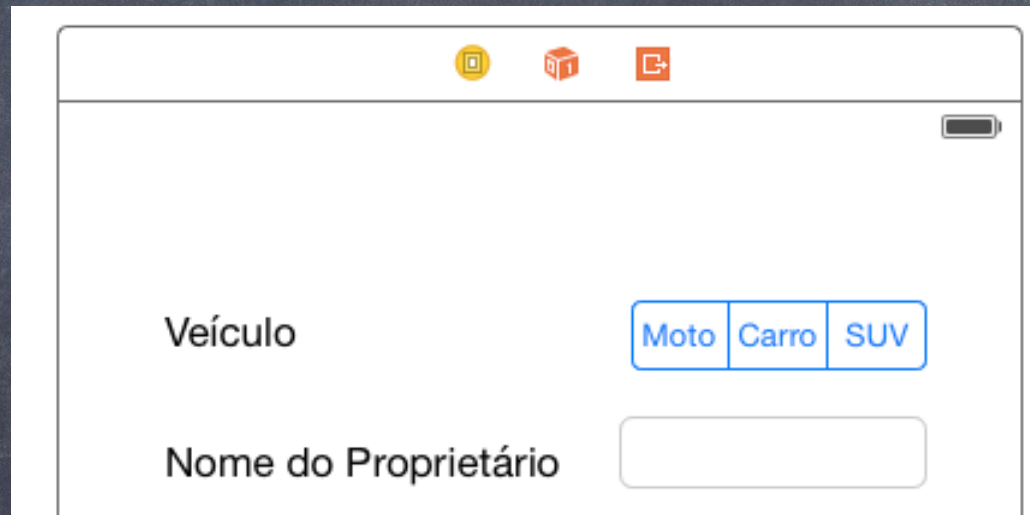
Segmented Control

- Clique no Attributes Inspector (1) e altere os dados conforme a imagem abaixo:



Label's e Text

- Inclua em sua tela dois Label's (Veículo e Nome do Proprietário) e um Text conforme a figura abaixo:



The image shows a UI mockup within a window frame. At the top of the window, there are three icons: a yellow circle with a white 'i', a red cube with a white '1', and a red square with a white 'G'. In the top right corner of the window, there is a battery status icon. The main content area contains two labels and a text input field. The first label is 'Veículo' and is positioned to the left of three blue buttons labeled 'Moto', 'Carro', and 'SUV'. The second label is 'Nome do Proprietário' and is positioned to the left of a white text input field with a thin grey border.

Stepper e Slider

- Inclua 4 Labels (Ano Modelo, KM mensal e dois label's) em frente de cada um, inclua também um Stepper para o Ano Modelo e um Slider para o KM mensal.

Veículo

Moto Carro SUV

Nome do Proprietário

Ano Modelo
Label

- +

KM mensal
Label

Alpha

Background

Tint

Drawing
☒ Opaque
☐ Hidden

☒ Clears Graphics Context

☐ Clip Subviews

☒ Autoresize Subviews

Stretching

X
Y

Width
Height

Text

Text Field - Displays editable text and sends an action message to a target object when Return is tapped.

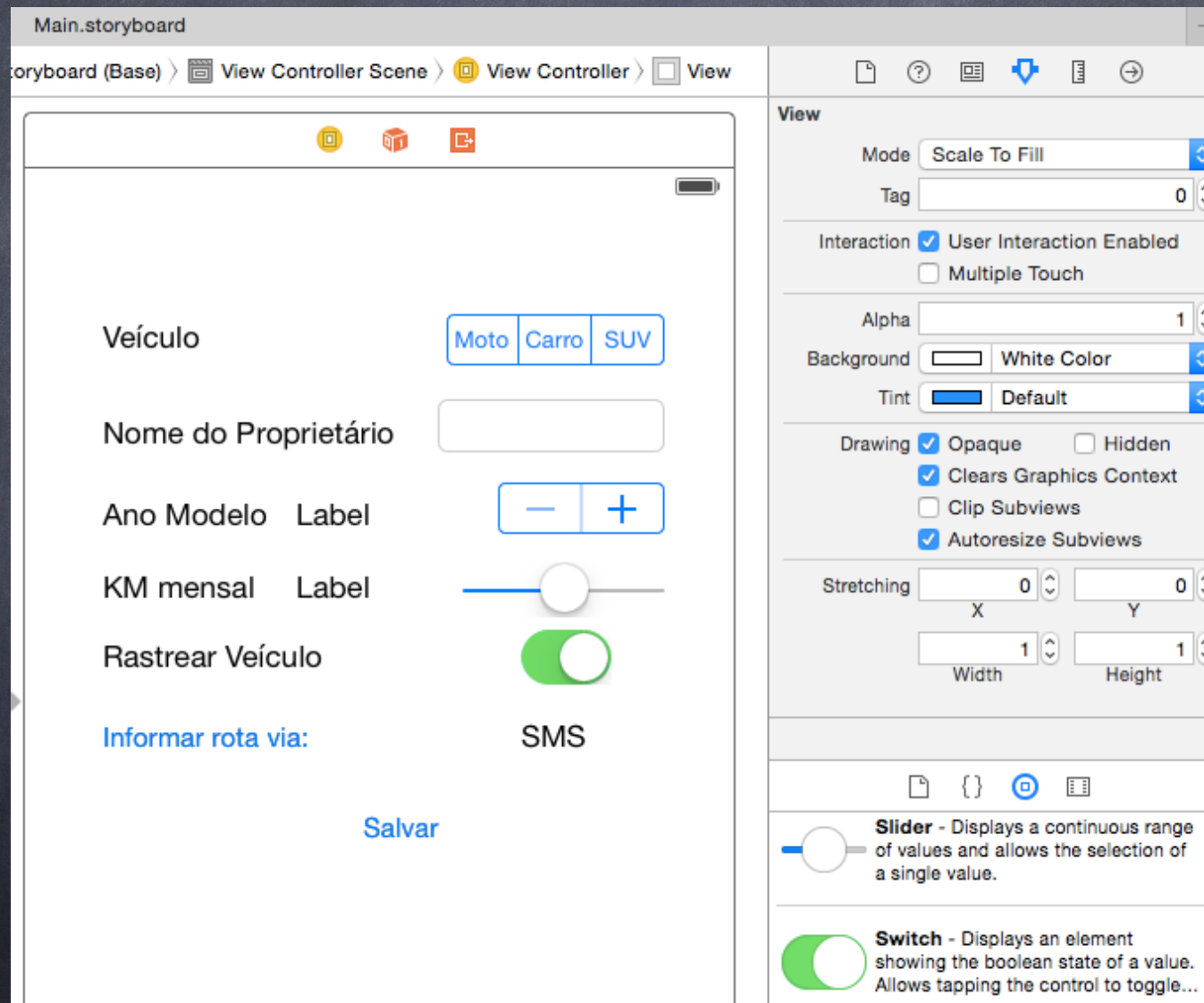
Slider - Displays a continuous range of values and allows the selection of a single value.

Switch - Displays an element showing the boolean state of a value. Allows tapping the control to toggle...

8

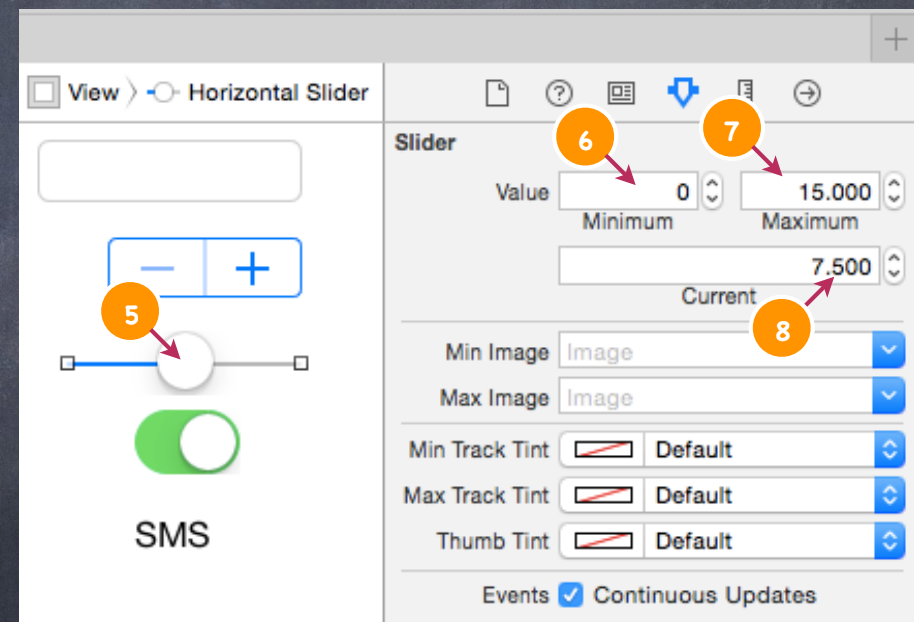
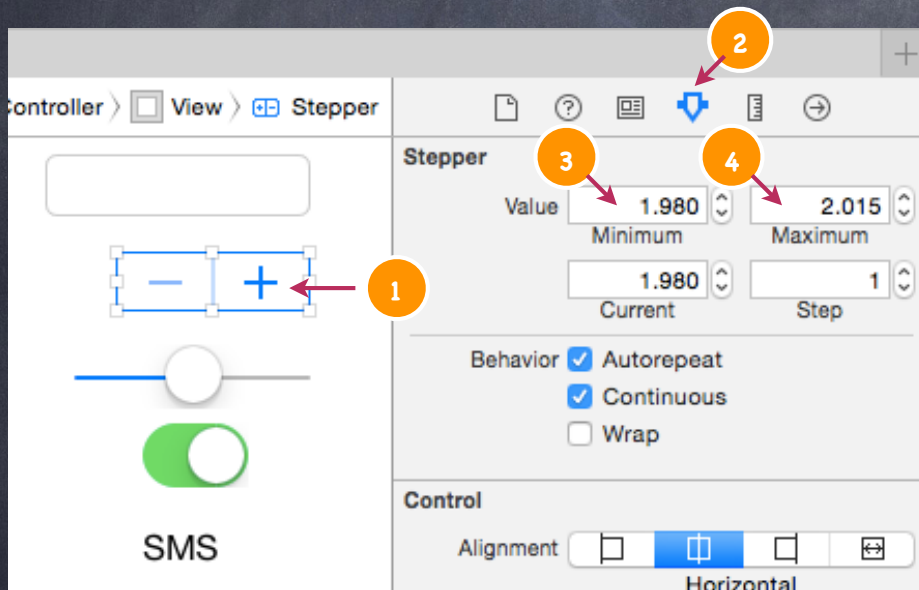
Switch

- Conforme a figura insira um Label Rastrear Veículo e um Switch, depois insira um botão com o texto: "Informar a rota via:" e em frente um outro Label escrito SMS, ao final insira um último botão com o texto Salvar.



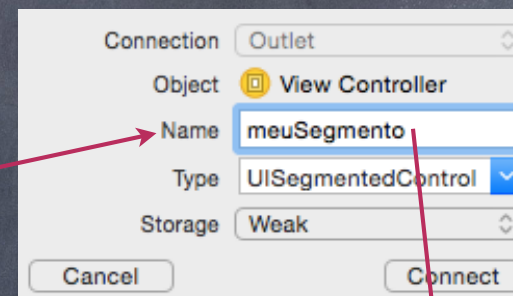
Stepper e Slider

- Selecione o Stepper e altere as propriedades Value Minimum e Value Maximum, depois selecione o Slider e altere as mesmas propriedades conforme a imagem abaixo:



Outlet's no viewController.swift

- No arquivo .swift crie os Outlet's dos componentes numerados, veja na imagem os seus respectivos nomes.



```

1 //
2 // ViewController.swift
3 // UIKit Swift
4 //
5 // Created by agesandro scarpioni on 14/02/16.
6 // Copyright © 2016 Agesandro Scarpioni. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController {
12
13     @IBOutlet weak var meuSegmento: UISegmentedControl!
14
15     @IBOutlet weak var lblAno: UILabel!
16
17     @IBOutlet weak var lblKm: UILabel!
18
19     @IBOutlet weak var lblRota: UILabel!
20
21     @IBOutlet weak var meuStepper: UIStepper!
22

```

Numbered callouts 1 through 5 are placed next to the corresponding outlet declarations in the code.

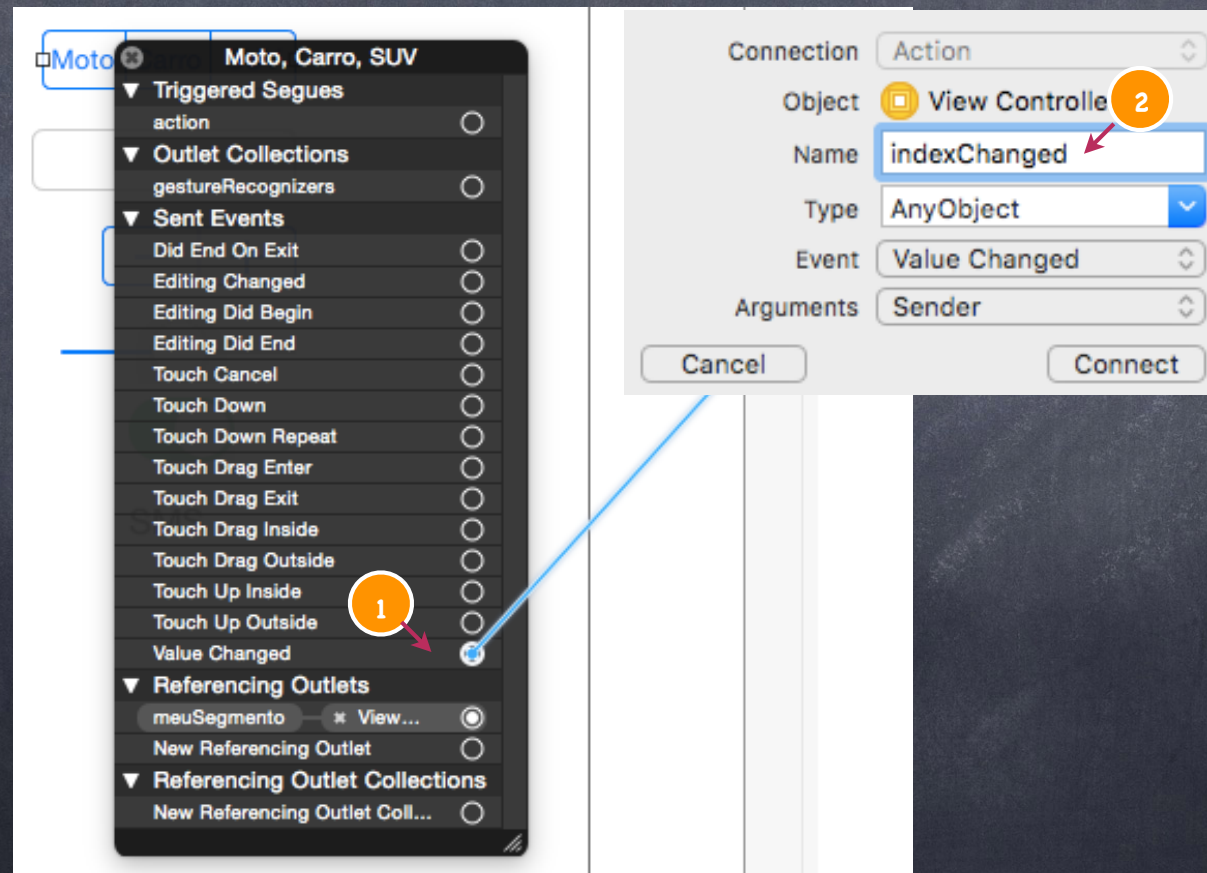
Arquivo .swift

- No método viewDidLoad escreva as linhas abaixo, ao executarmos o programa as informações nos label's irão aparecer.

```
23  
24     override func viewDidLoad() {  
25         super.viewDidLoad()  
26         lblAno.text = "1980"  
27         lblKm.text = "7500"  
28     }  
29
```


IBAction do Segmented Control

- Clique com o botão direito sobre o Segmented Control escolha Value Changed(1), arraste para dentro da área da class ViewController.swift, em Name (2) digite indexChanged.



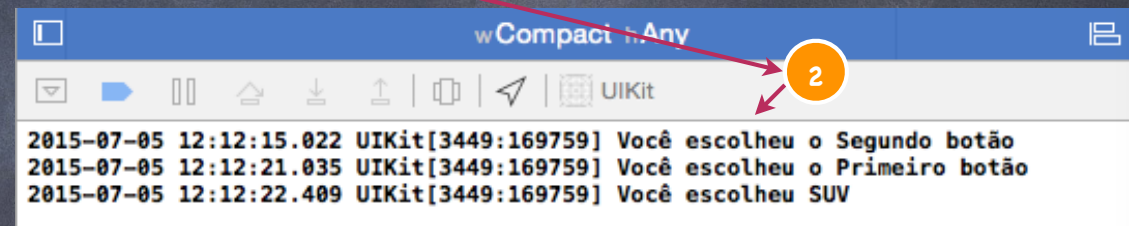
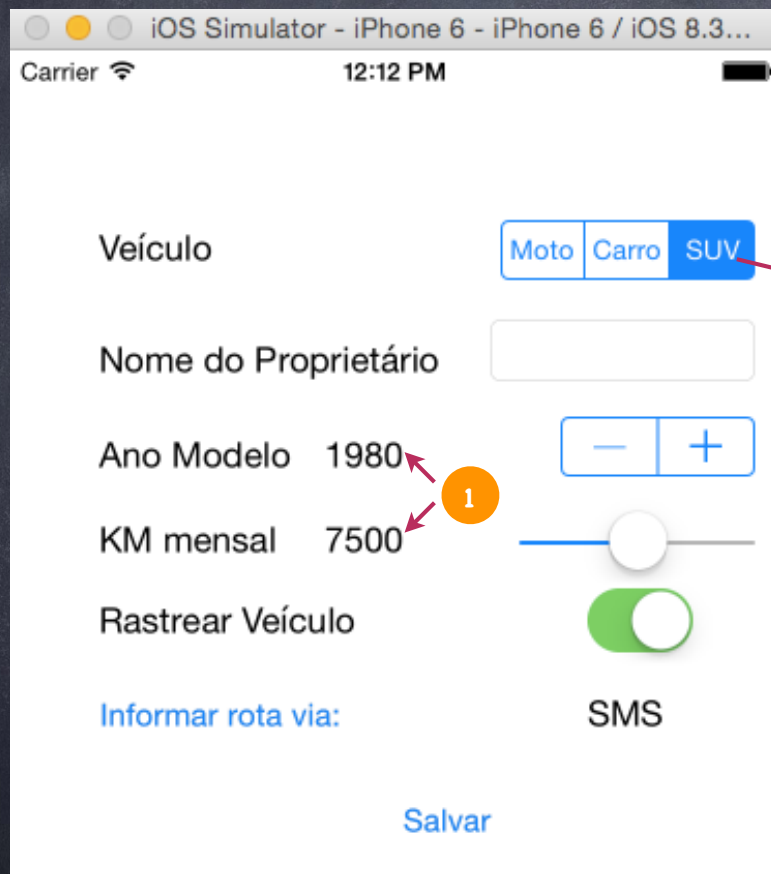
Segmented Control

- Para recuperar qual botão foi clicado usamos a propriedade `selectedSegmentIndex`, podemos fazer com comando `if` ou `switch`, optei por fazer com `switch`, observe que na terceira opção (case 2) decidi mostrar o texto do `SegmentedControl`, para recuperarmos o tal texto usamos o método `titleForSegmentAtIndex` e informamos o `selectedSegmentIndex`.

```
26
27 @IBAction func indexChanged(sender: AnyObject) {
28
29     switch (meuSegmento.selectedSegmentIndex){
30     case 0:
31         print("Você escolheu o primeiro botão")
32         break
33     case 1:
34         print("Você escolheu o segundo botão")
35         break
36     case 2:
37         print("Você escolheu o \(self.meuSegmento.
38             titleForSegmentAtIndex(meuSegmento.
39                 selectedSegmentIndex)!)")
40         break
41     default:
42         break
43     }
44 }
```

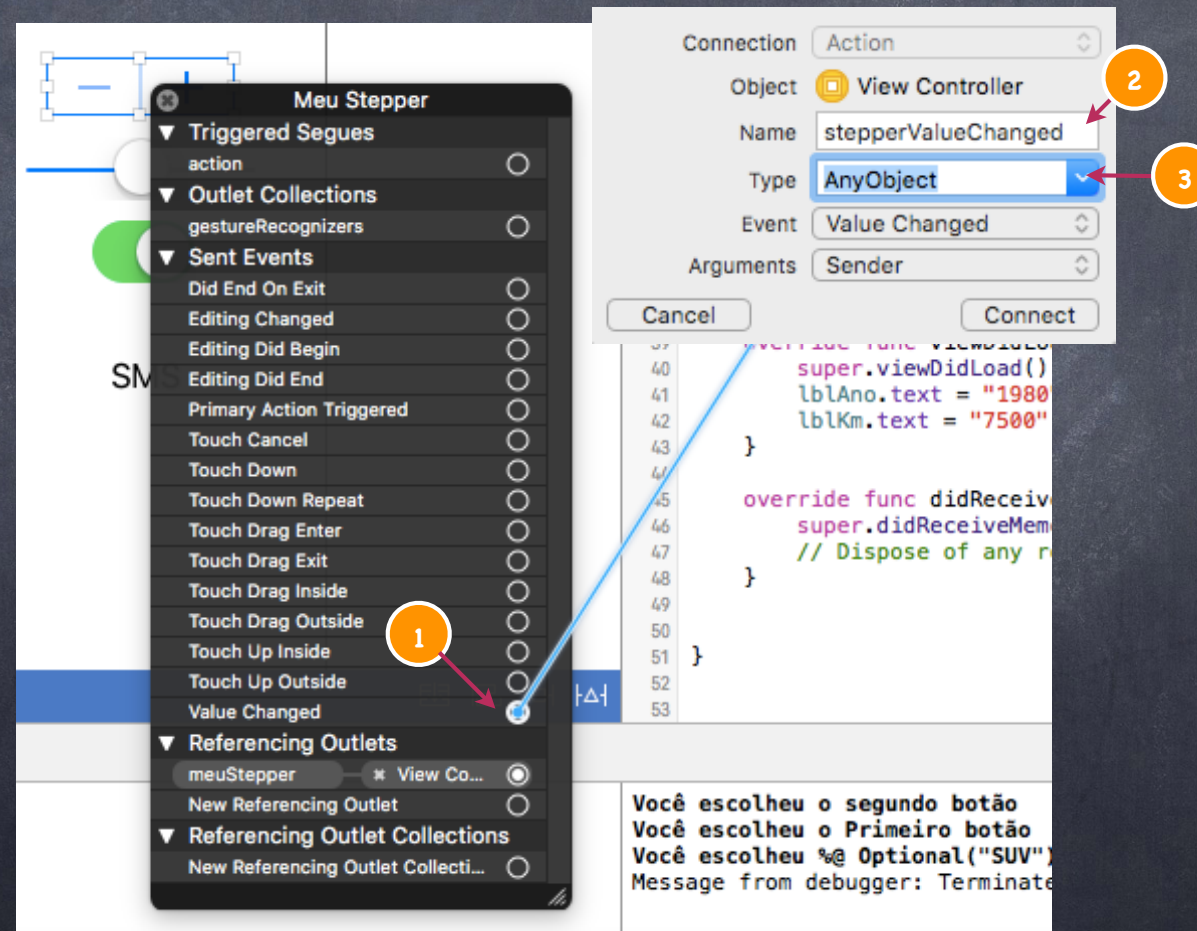

Executando o App

- Command + R para executar, veja as informações que já aparecem nos label's (1) viaDidLoad e os resultados do NSLog no Console (2) clicando nos botões Carro, Moto e SUV.



IBAction no Arquivo .swift do Stepper

- Clique com o botão direito sobre o Stepper, escolha Value Changed(1), em Name (2) digite stepperValueChanged em Type escolha AnyObject (3).



Implementação do Stepper

- A implementação do Stepper é muito simples, basta passar o value do stepper para o lblAno, porém, como o lblAno é texto e o value do Stepper é um Double, temos que utilizar o Int para que não apareça casas decimais, como ainda é um número precisamos converter para String utilizando "\\".
- Lembre-se que meuStepper é o nome do Outlet que demos para o Stepper e lblAno foi o nome do Outlet que demos para o Label, stepperValueChanged é o nome que demos para o IBAction do Stepper.

```
38  
39 @IBAction func stepperValueChanged(sender: AnyObject) {  
40     lblAno.text = "\\"(Int(meuStepper.value))"  
41 }  
42
```


Implementação do Stepper

- Uma outra possibilidade é não criar o Outlet com o nome de meuStepper, e ao criarmos o Action ao invés de selecionarmos no Type o tipo AnyObject selecionamos UIStepper e a assinatura ficaria da seguinte forma:

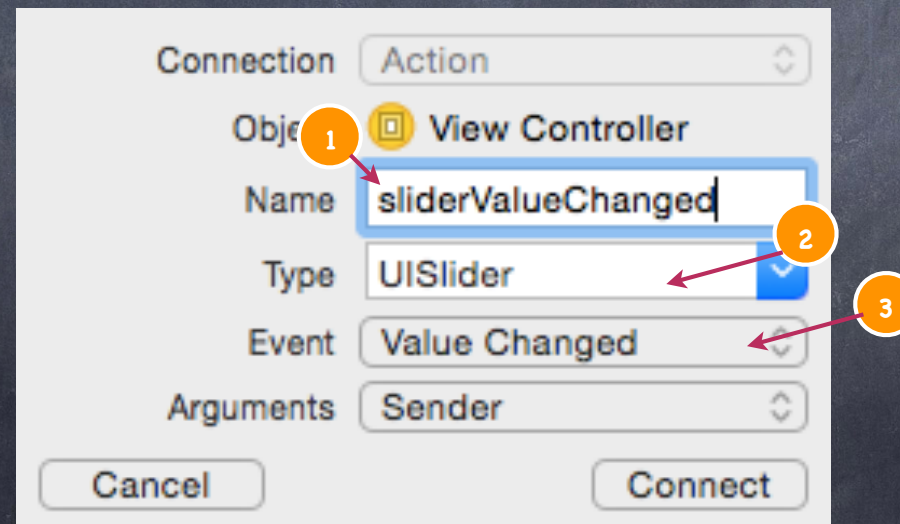
```
41  
42     @IBAction func stepperValueChanged(sender: UIStepper) {  
43         lblAno.text = "\(Int(sender.value))"  
44     }  
45
```

- Compare essas duas imagens, abaixo usamos meuStepper.value, na imagem acima utilizamos sender.value, faremos da segunda forma com o objeto Slider.

```
38  
39     @IBAction func stepperValueChanged(sender: AnyObject) {  
40         lblAno.text = "\(Int(meStepper.value))"  
41     }  
42
```


IBAction do Slider

- 1 Clique com o botão direito sobre o Slider, escolha Value Changed leve até o arquivo.swift, em Name (1) digite sliderValueChanged, troque o type (2) de ID para UISlider.
- 2 Quando trocamos o tipo, não precisamos criar um Outlet do componente para enviarmos o value, veja a programação no próximo slide e compare com a programação no slide anterior onde o tipo era AnyObject.

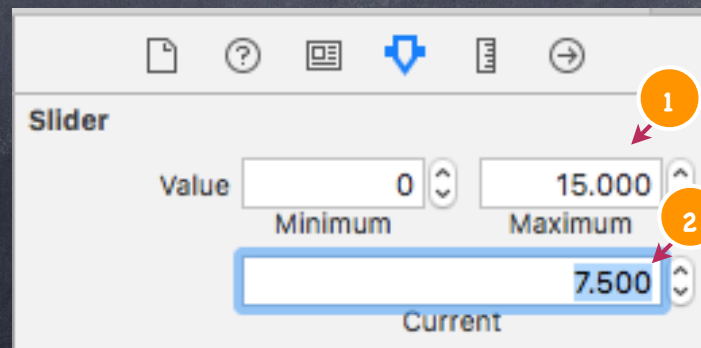


Implementação do Slider

- A implementação do Slider é simples como o Stepper, basta ir no arquivo.swift e passar o value do Slider para o lblKm, veja que como mudamos o type, ao invés de utilizarmos o nome de um Outlet para o slider usamos sender.value.

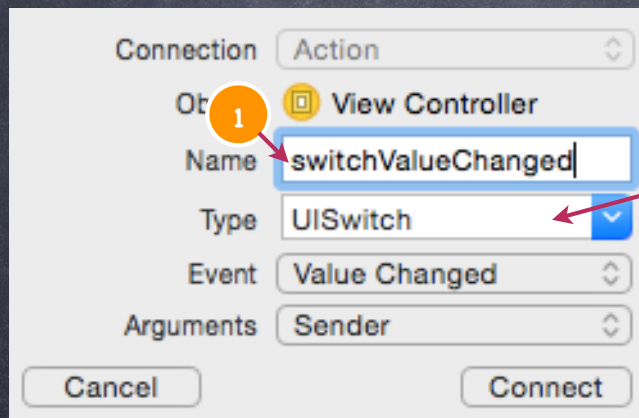
```
53  
54 @IBAction func sliderValueChanged(sender: UISlider) {  
55     lblKm.text = "\(sender.value)"  
56 }  
57
```

- Lembre-se que o intervalo do slider foi alterado para o máximo 15000, já o Current foi ajustado para 7500, isso ocorreu para posicionar o botão do slider no centro.



IBAction do Switch no arquivo.swift

- Clique com o botão direito sobre o Switch, escolha Value Changed leve até o arquivo.swift, em Name (1) digite switchValueChanged, troque o type (2) de ID para UISwitch. Depois vá até o arquivo.m (3) e implemente, execute o programa e veja o console (4), transforme o sender para inteiro (5) e veja o resultado (6).



```
58  
59 @IBAction func switchValueChanged(sender: UISwitch) {  
60     print(sender.on)  
61 }
```

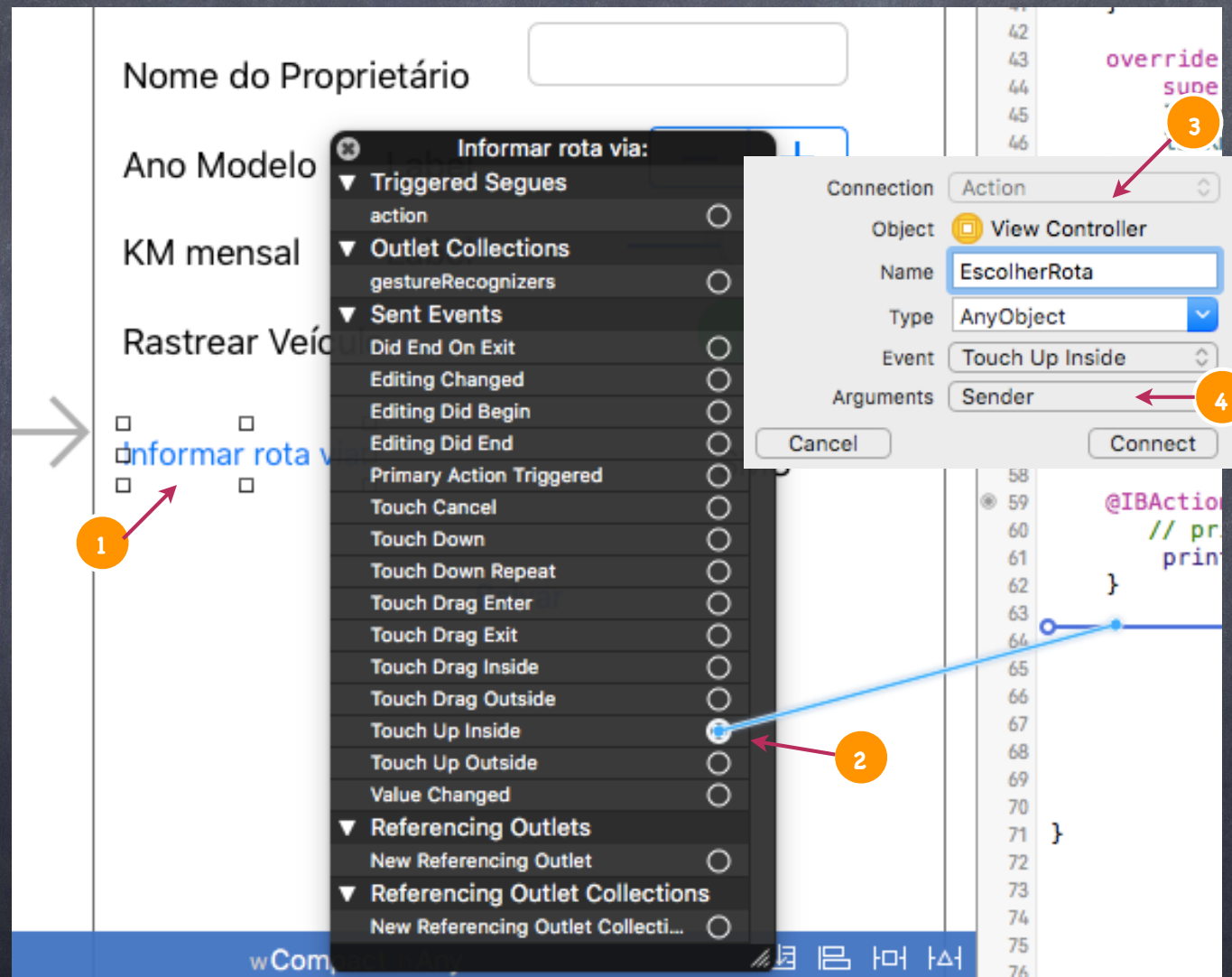
```
false  
true  
  
All Output
```

```
58  
59 @IBAction func switchValueChanged(sender: UISwitch) {  
60     // print(sender.on)  
61     print(Int(sender.on))  
62 }
```

```
0  
1  
  
All Output
```


IBAction do botão

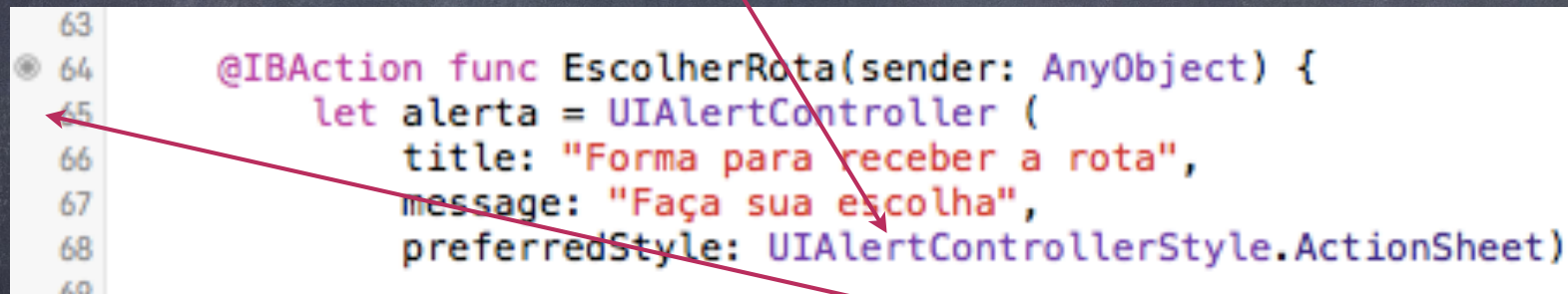
"informar rota via" no arquivo.swift



Implementação

- Inicie a implementação da IBAction EscolherRota, nas linhas abaixo vamos criar uma constante chamada alerta. Para informar o .ActionSheet você deverá dar 2 cliques na palavra UIAlertControllerStyle e depois digitar o .ActionSheet, ou apenas digite .ActionSheet.

```
63
64 @IBAction func EscolherRota(sender: AnyObject) {
65     let alerta = UIAlertController (
66         title: "Forma para receber a rota",
67         message: "Faça sua escolha",
68         preferredStyle: UIAlertControllerStyle.ActionSheet)
69
```



- Não se preocupe com o Warning que irá aparecer nesse ponto, ao continuar o código no próximo slide esse aviso irá desaparecer

Implementação (Continuação)

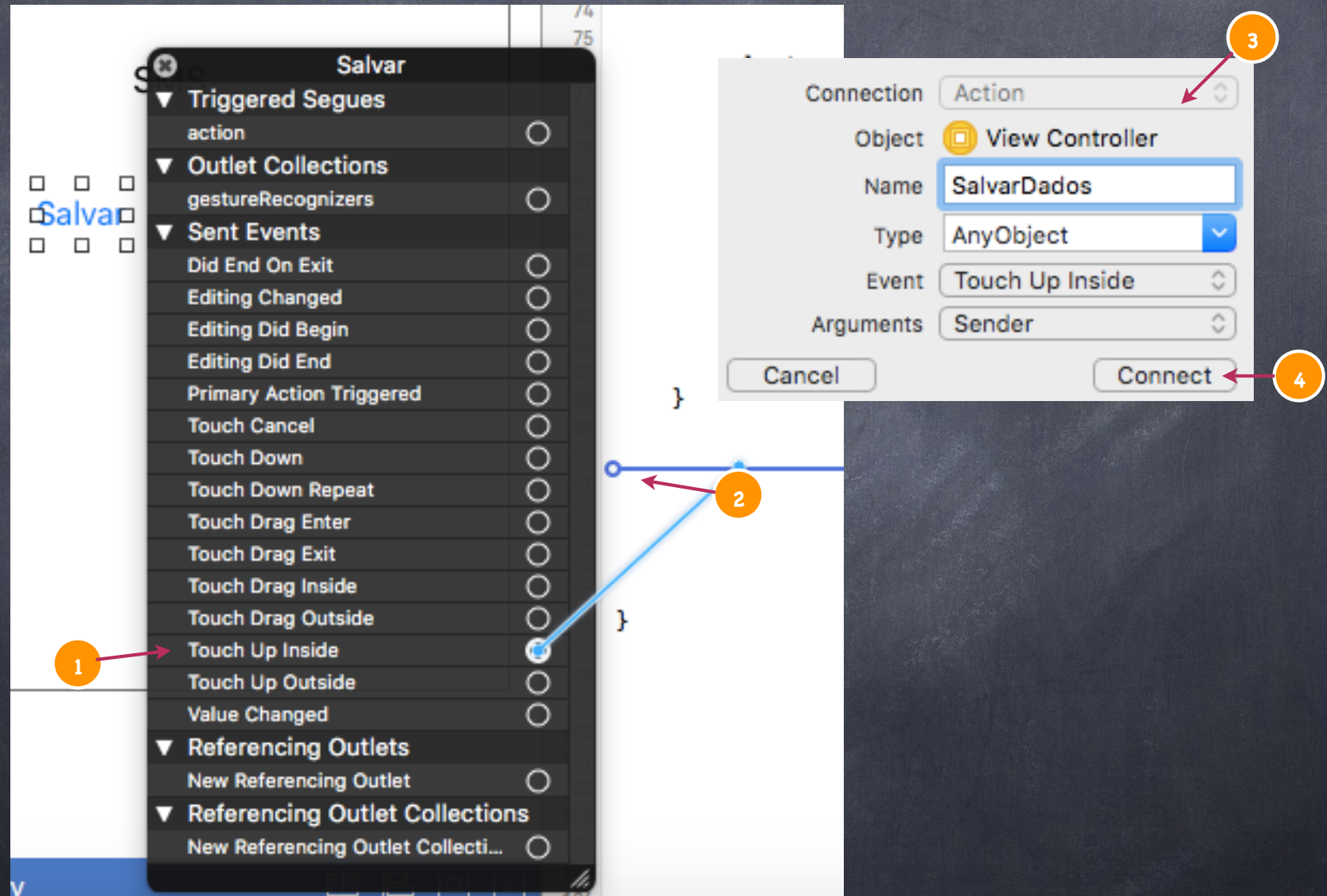
```
63
64 @IBAction func EscolherRota(sender: AnyObject) {
65     let alerta = UIAlertController (
66         title: "Forma para receber a rota",
67         message: "Faça sua escolha",
68         preferredStyle: UIAlertControllerStyle.ActionSheet)
69
70     alerta.addAction(UIAlertAction(
71         title: "SMS",
72         style: UIAlertActionStyle.Default,
73         handler: { action in
74             self.lblRota.text = action.title}))
75
76     alerta.addAction(UIAlertAction(
77         title: "E-mail",
78         style: UIAlertActionStyle.Default,
79         handler: { action in
80             self.lblRota.text = action.title}))
81
82     alerta.addAction(UIAlertAction(
83         title: "Cancelar",
84         style: UIAlertActionStyle.Cancel,
85         handler:nil))
86
87     presentViewController(alerta, animated: true, completion: nil)
88
89 }
90
91
```


Execute - Command + R



Obs: Veja que esse tipo de mensagem (ActionSheet) é exibida na parte inferior da tela do iPhone.

IBAction do botão "Salvar" no arquivo.swift

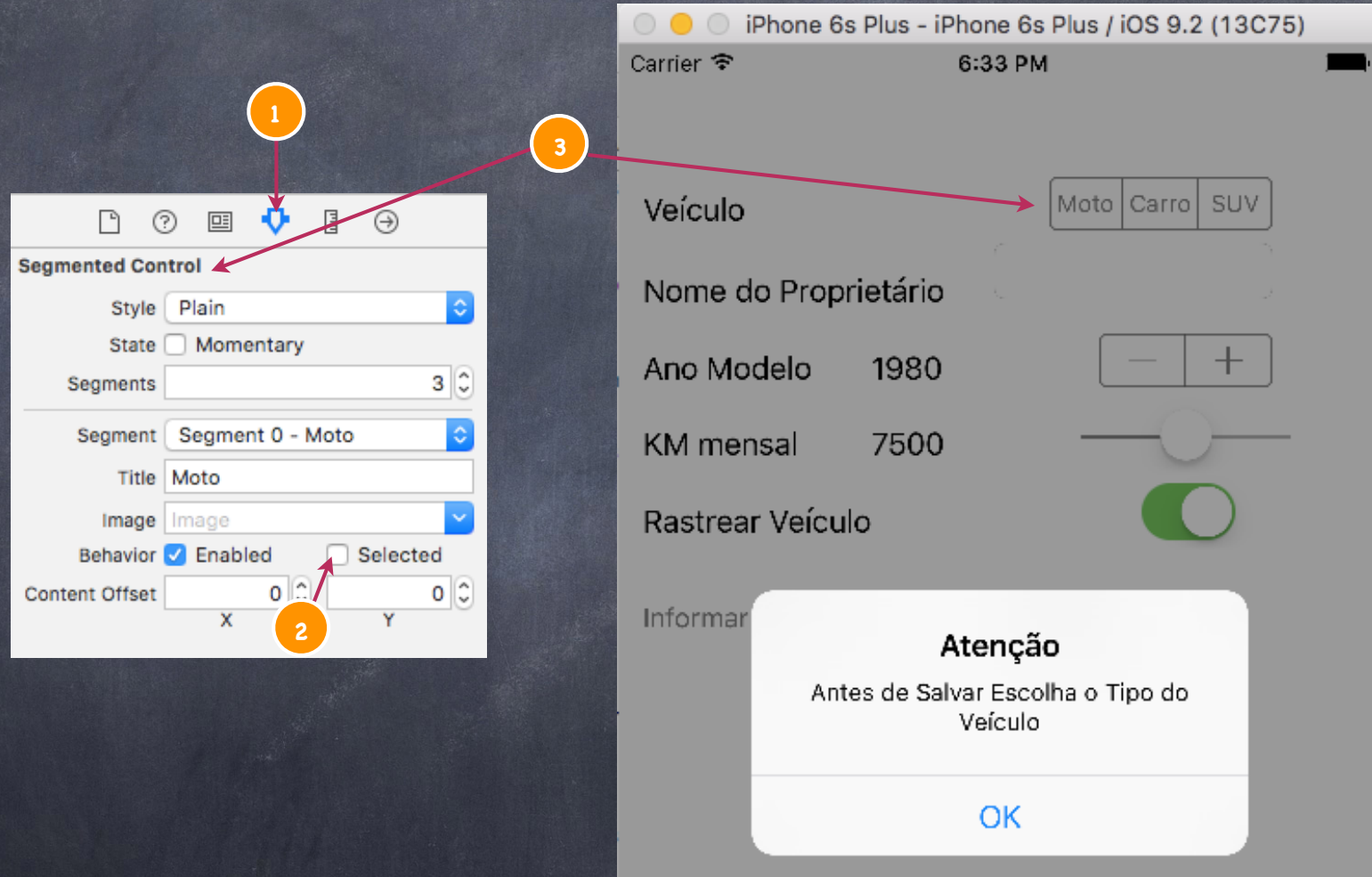


Implementação do Botão Salvar

```
90
91 @IBAction func SalvarDados(sender: AnyObject) {
92     var msg: String
93     var tipo: String
94
95     if meuSegmento.selectedSegmentIndex >= 0{
96         tipo = meuSegmento.titleForSegmentAtIndex(meuSegmento.selectedSegmentIndex)!
97         msg = "Veículo \(tipo) do ano \(lblAno.text!) salvo com sucesso"
98     }else{
99         tipo = ""
100         msg = "Antes de Salvar Escolha o Tipo do Veículo"
101
102     }
103
104     let alerta = UIAlertController (
105         title: "Atenção",
106         message: msg,
107         preferredStyle: UIAlertControllerStyle.Alert)
108
109     alerta.addAction(UIAlertAction(
110         title: "OK",
111         style: UIAlertActionStyle.Default,
112         handler: nil))
113
114
115     presentViewController(alerta, animated: true, completion: nil)
116
117 }
118
119
```


Command + R

- Em Attributes Inspector(1), desmarque a opção Selected(2) do Segmented Control(3) para que nosso IF funcione.



Obs: Veja que esse tipo de mensagem (Alert) é exibida na centro da tela do iPhone.