

App Agenda

parte 1 – Desenhando e preparando a Interface

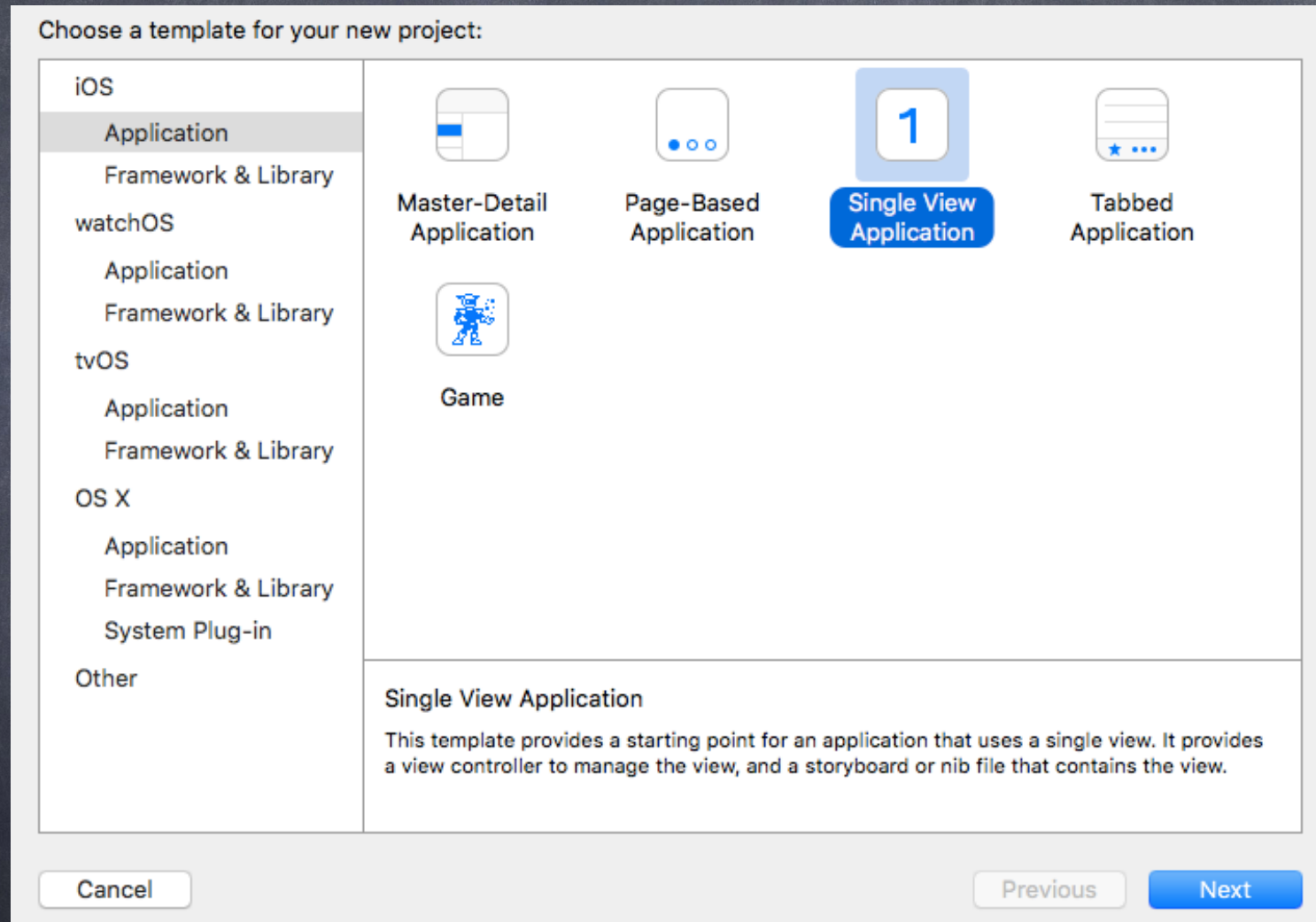
X-Code com Swift
Prof. Agesandro Scarpioni

App - Agenda

- Criar um aplicativo para que se digite um nome, sobrenome e telefone. Aprender a manipular o teclado e no futuro este app será utilizado as aulas de persistência.

Iniciando o Projeto

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



OBS: Este é o template que já cria uma viewController e uma classe com o arquivo.swift, inclusive o delegate já tem a referência dessa classe.

Caixas de Texto

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

Choose options for your new project:

Product Name:

Organization Name:

Organization Identifier:

Bundle Identifier:

Language:

Devices:

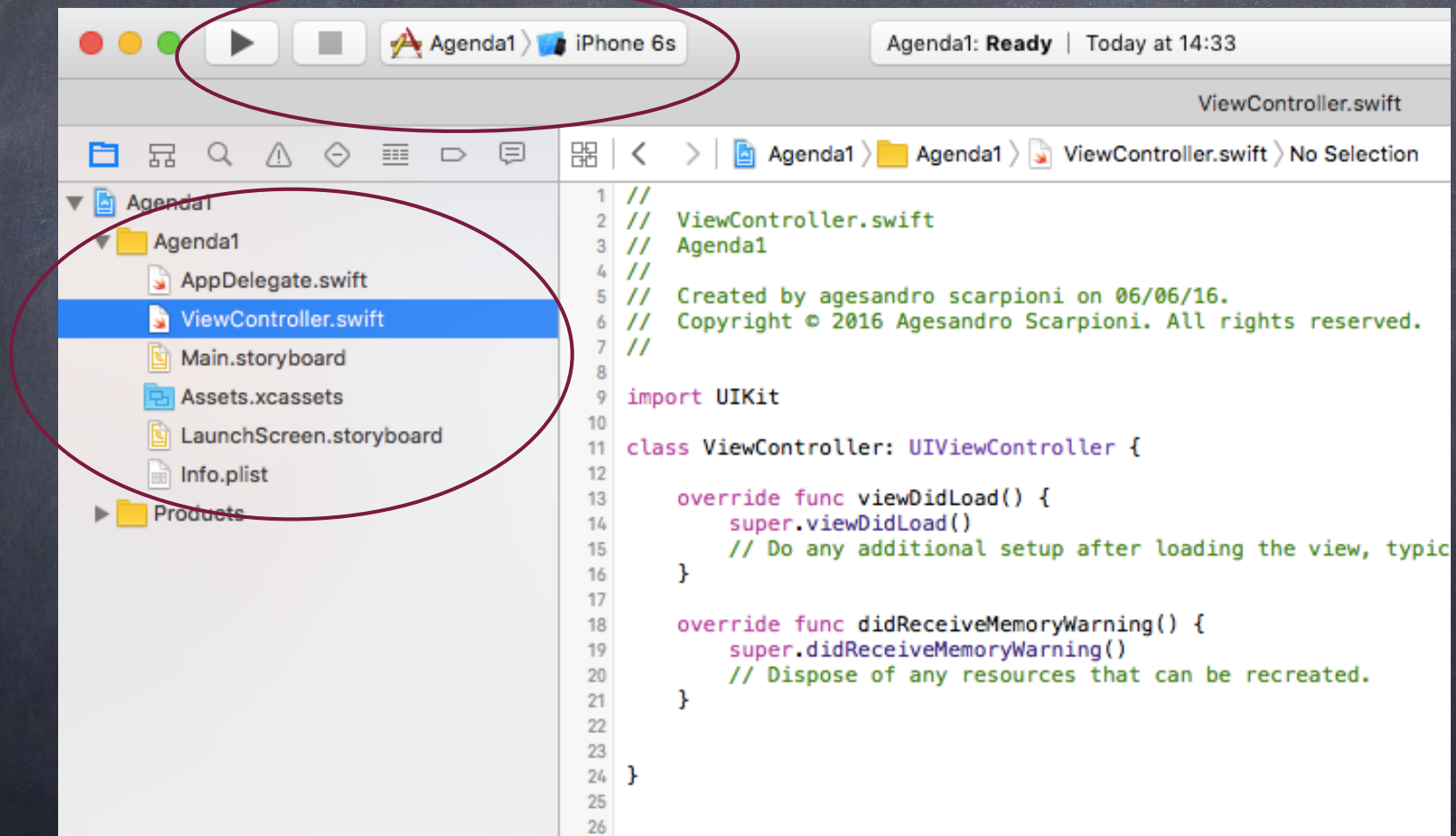
☐ Use Core Data

☐ Include Unit Tests

☐ Include UI Tests

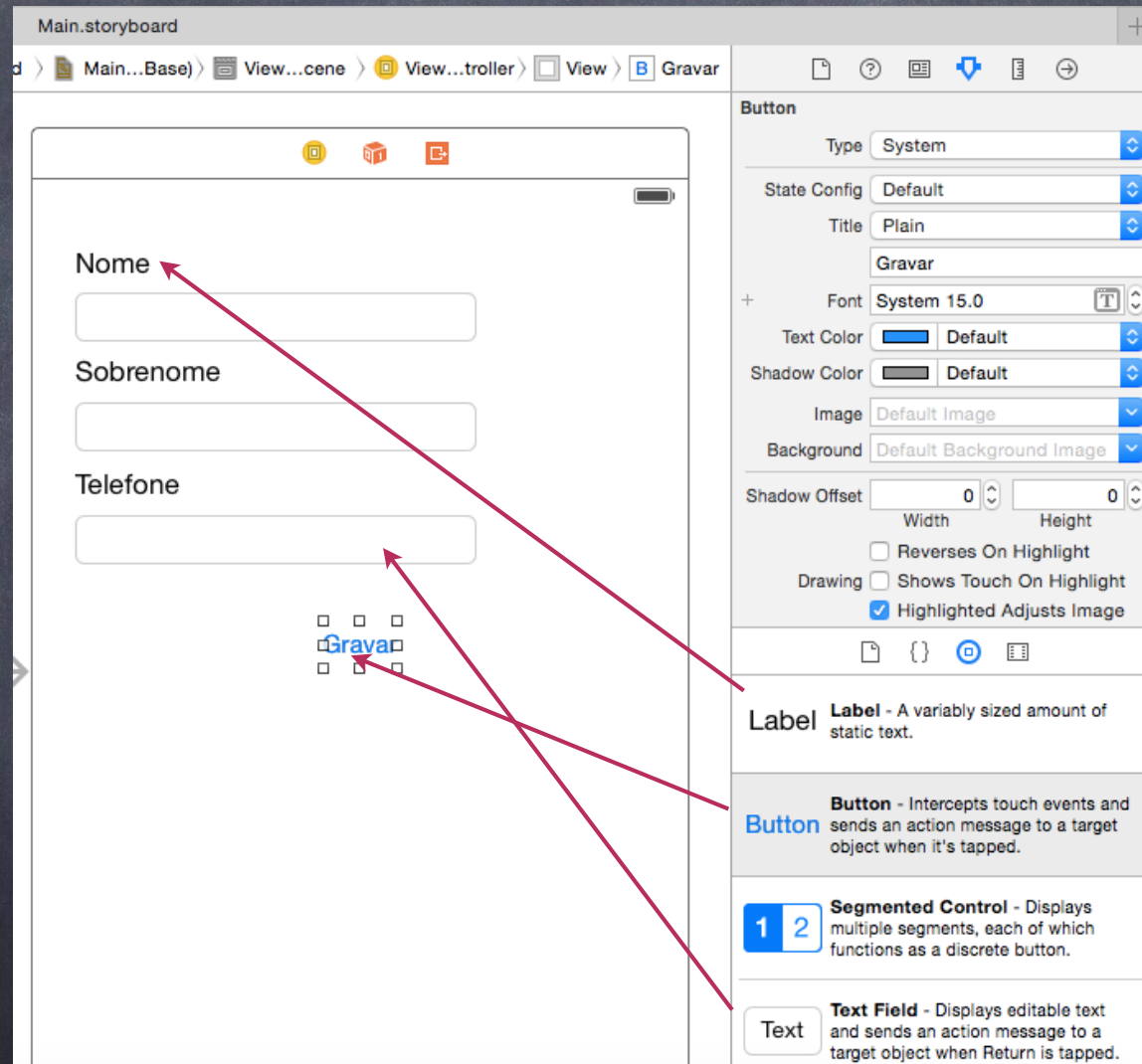
Tudo em seu lugar

- Note que utilizando este template foi incluída uma classe chamada ViewController.swift, o delegate que já possui a classe referenciada e um Storyboard.



Adicionando objetos

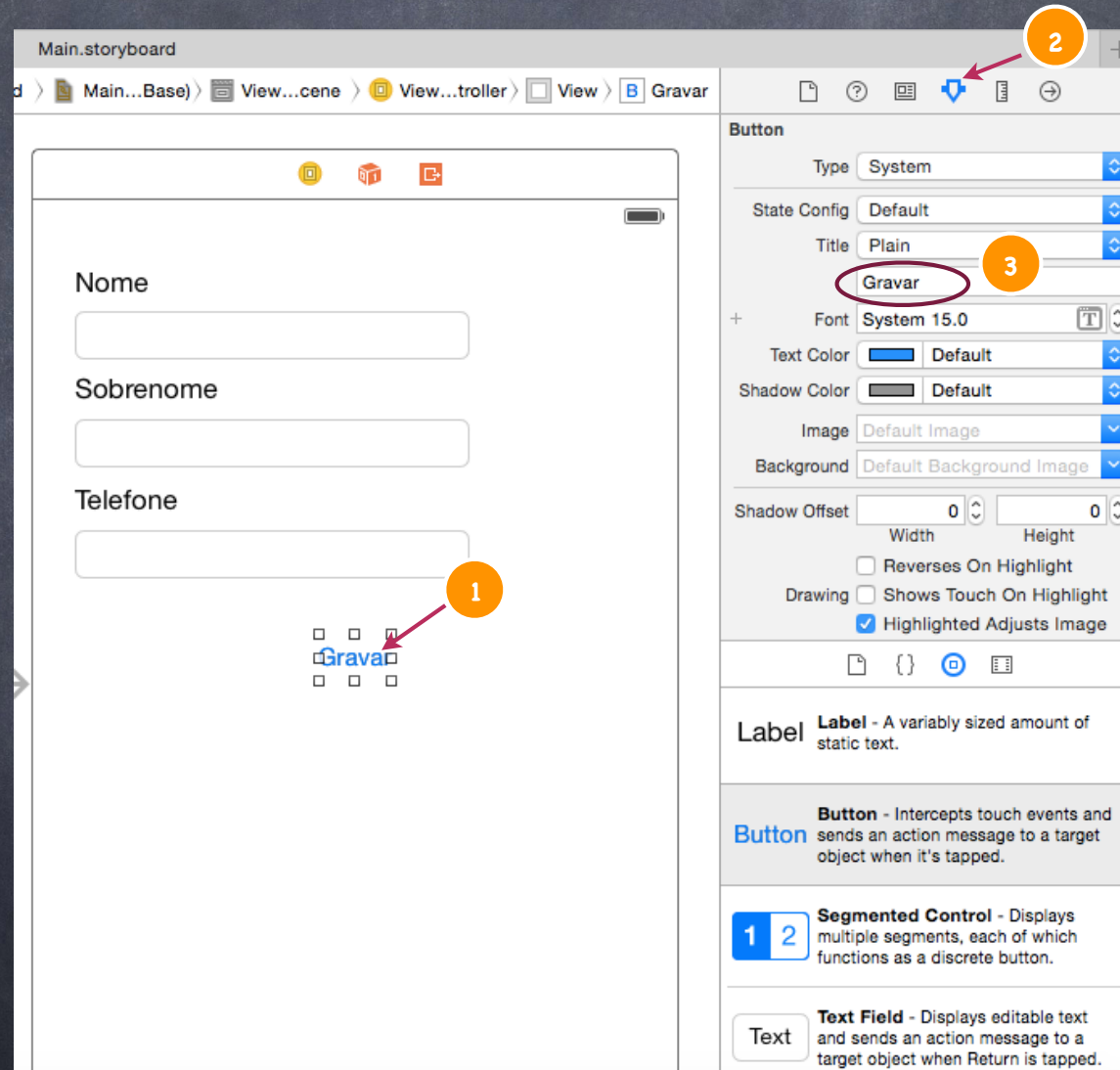
- Inclua 3 label's, 1 Button e 3 Text Field's na Storyboard do iPhone, altere o texto do botão para Gravar, e para os label's coloque Nome, Sobrenome, Telefone.



Dica: Se você esqueceu como trocar os textos veja no próximo slide ou dê dois cliques no objeto.

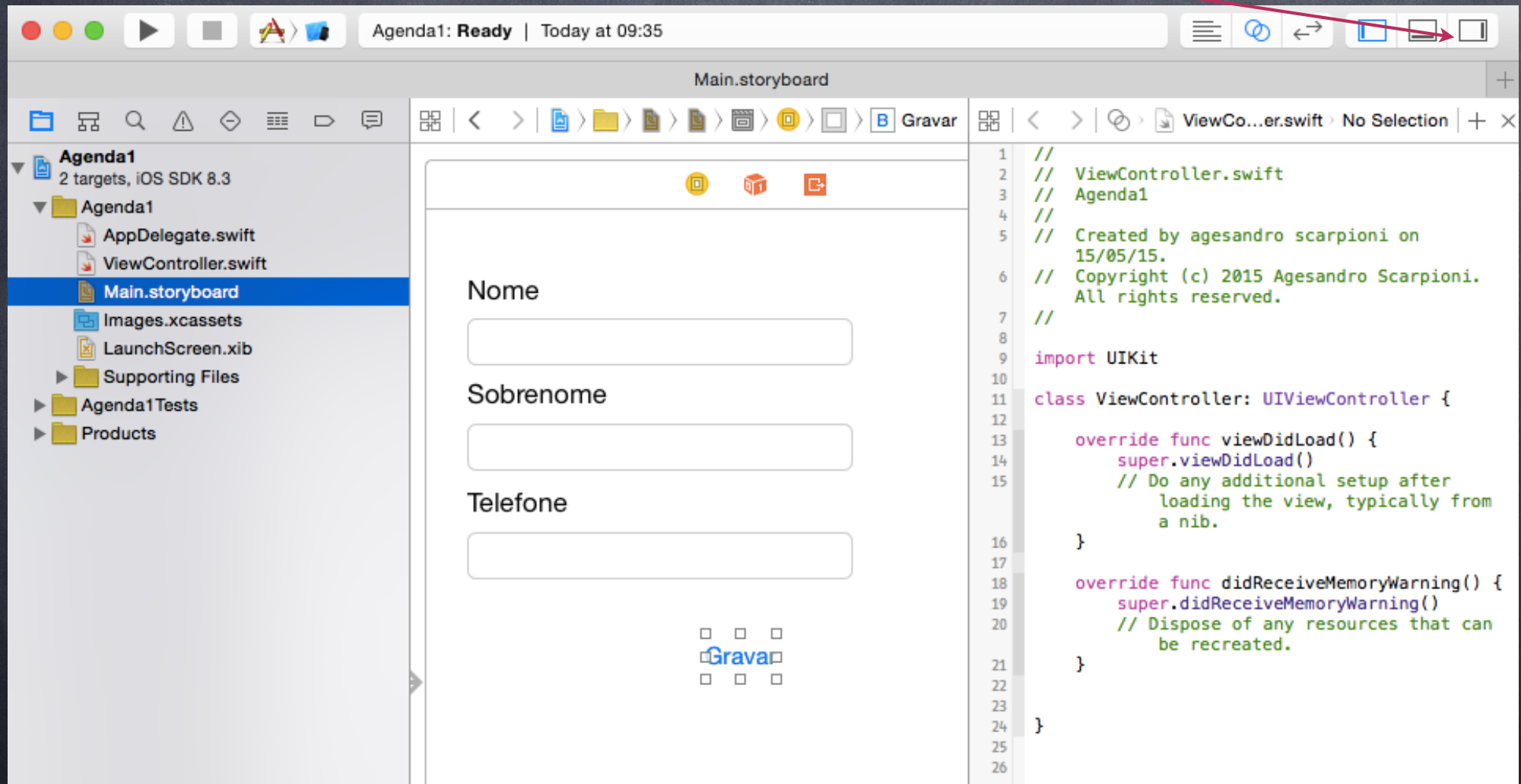
Alterando os textos

- Basta clicar no label ou no botão (item 1), verifique se você está em Show Attributes Inspector (item 2) e altere o texto para Gravar (item 3).



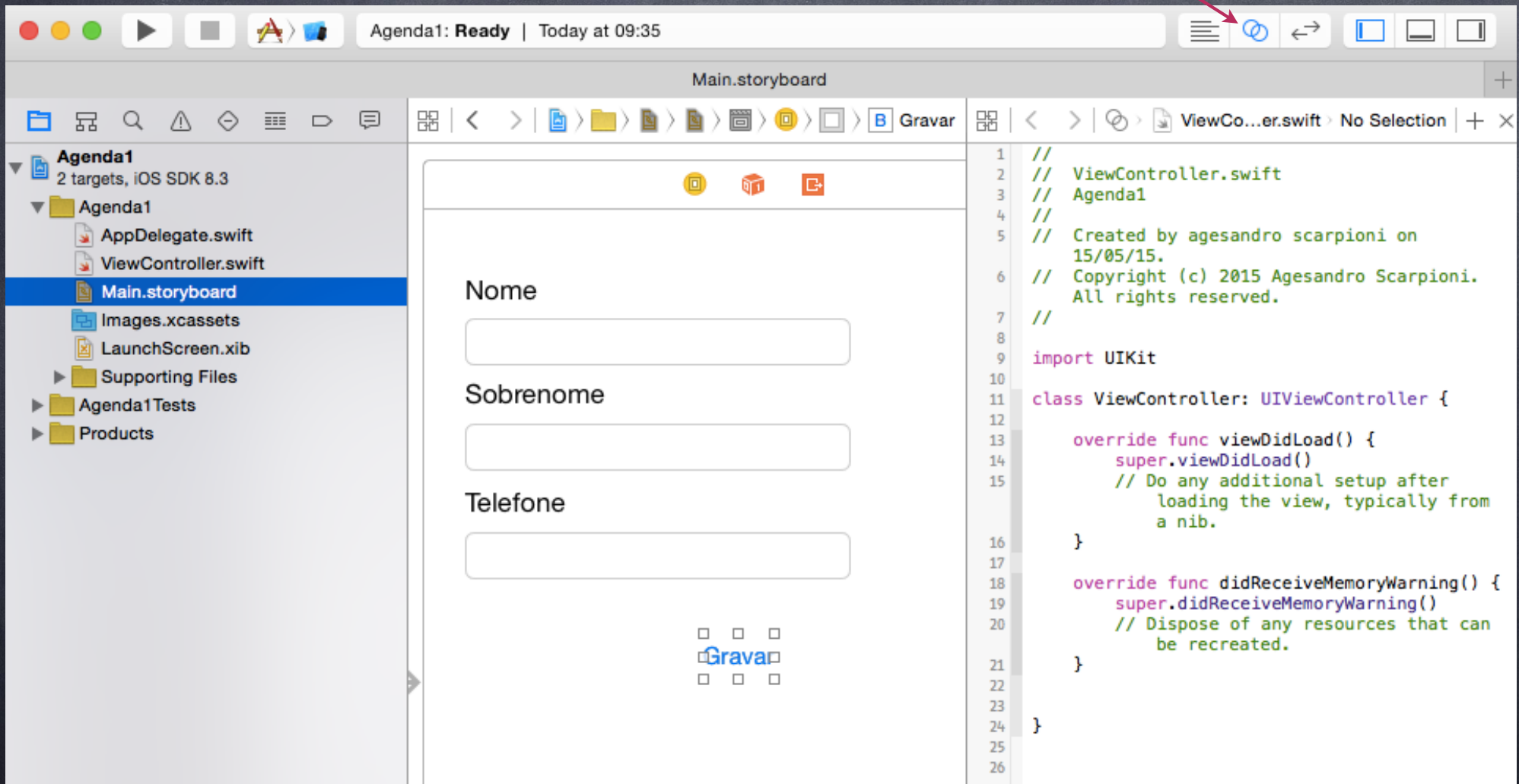
Declarando os outlet's

- Primeiramente organize o ambiente, esconda a janela da esquerda e clique neste ícone.



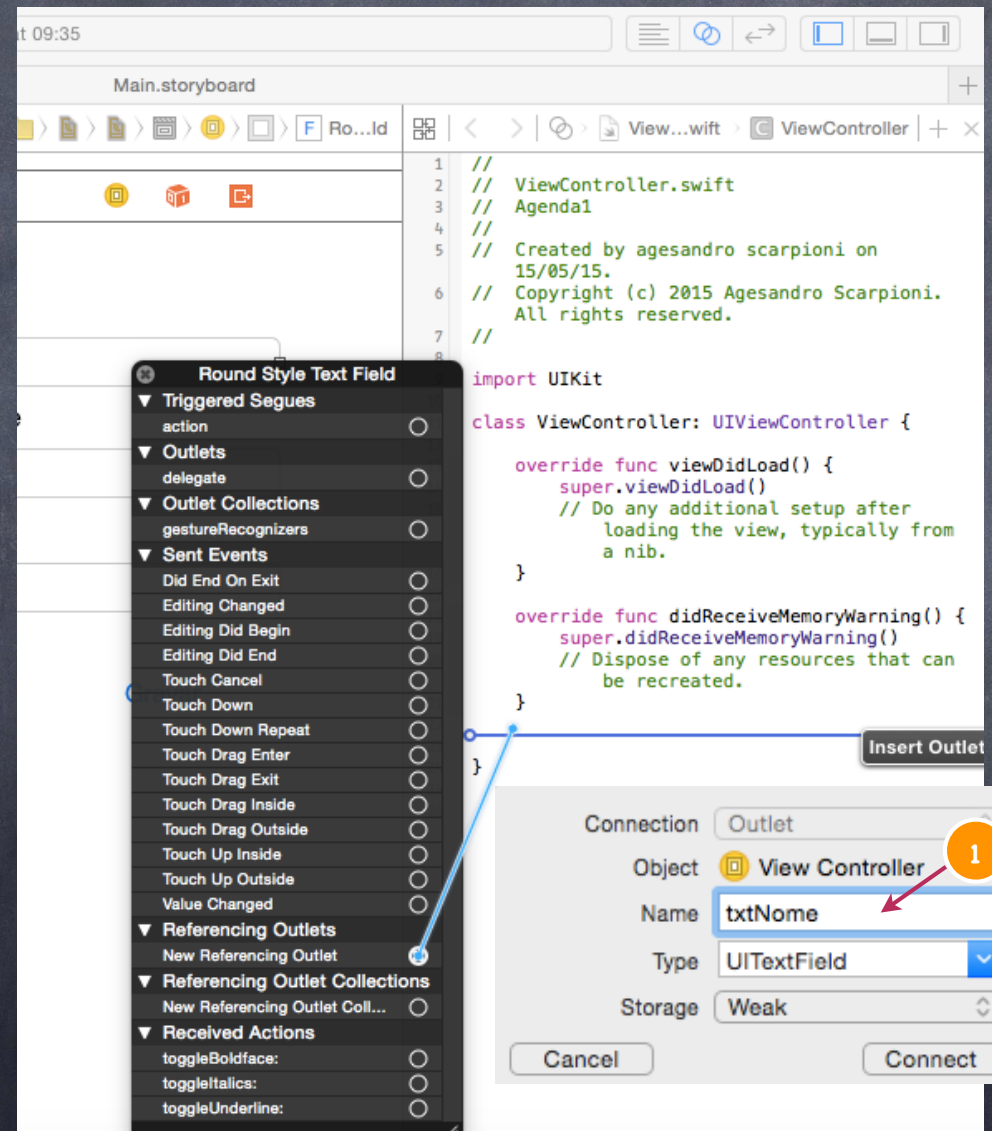
Declarando os outlet's

- Compartilhe a tela de Storyboard com a viewController.swift obtendo as duas telas simultaneamente clicando neste ícone.



Declarando os outlet's

- Clique com o botão direito sobre o 1º text field e escolha "New Referencing Outlet's" clicando no local indicado na figura e arrastando até a área acima da última chave. Ao aparecer a janela (1), nomeie como txtNome e clique em Connect.



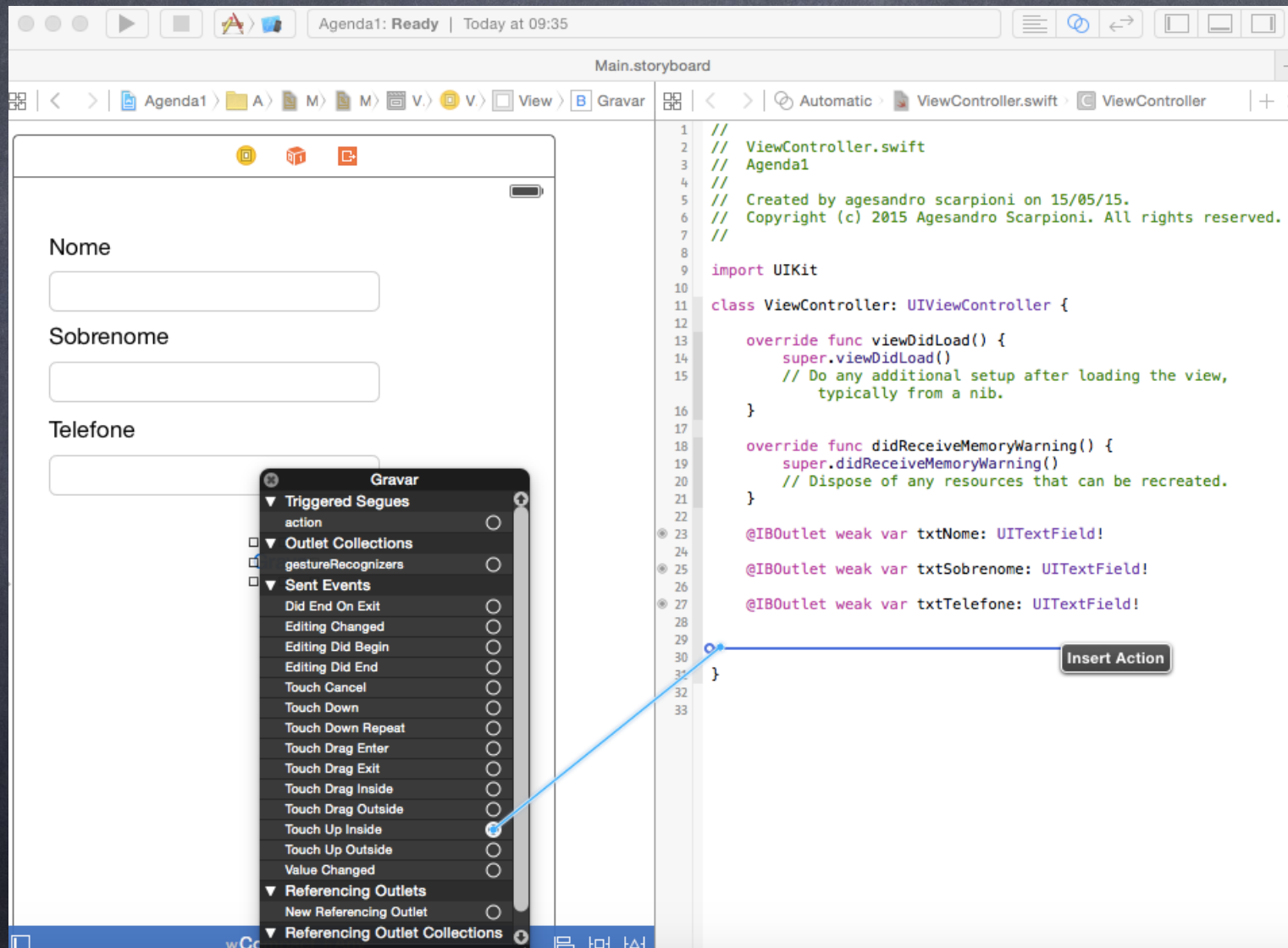
Declarando os outlet's

- Repita o mesmo para os outros dois text's, ao final essas três linhas (IBOutlet's) são declaradas automaticamente, e os três text's da View também já estão relacionados aos seus respectivos outlet's.

```
1 //
2 // ViewController.swift
3 // Agenda1
4 //
5 // Created by agesandro scarpioni on 15/05/15.
6 // Copyright (c) 2015 Agesandro Scarpioni. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController {
12
13     override func viewDidLoad() {
14         super.viewDidLoad()
15         // Do any additional setup after loading the view,
16         // typically from a nib.
17     }
18
19     override func didReceiveMemoryWarning() {
20         super.didReceiveMemoryWarning()
21         // Dispose of any resources that can be recreated.
22     }
23
24     @IBOutlet weak var txtNome: UITextField!
25
26     @IBOutlet weak var txtSobrenome: UITextField!
27
28     @IBOutlet weak var txtTelefone: UITextField!
29
30 }
```

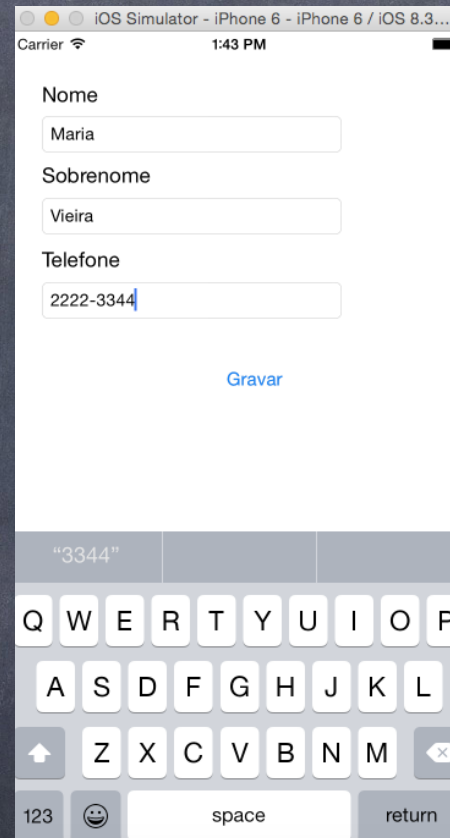

Declarando IBAction automaticamente

- Clique com o botão direito do mouse sobre o botão Gravar, escolha o evento "Touch Up Inside" e arraste para a área abaixo dos Outlet's, nomeie como btnGravar.



Execute e observe

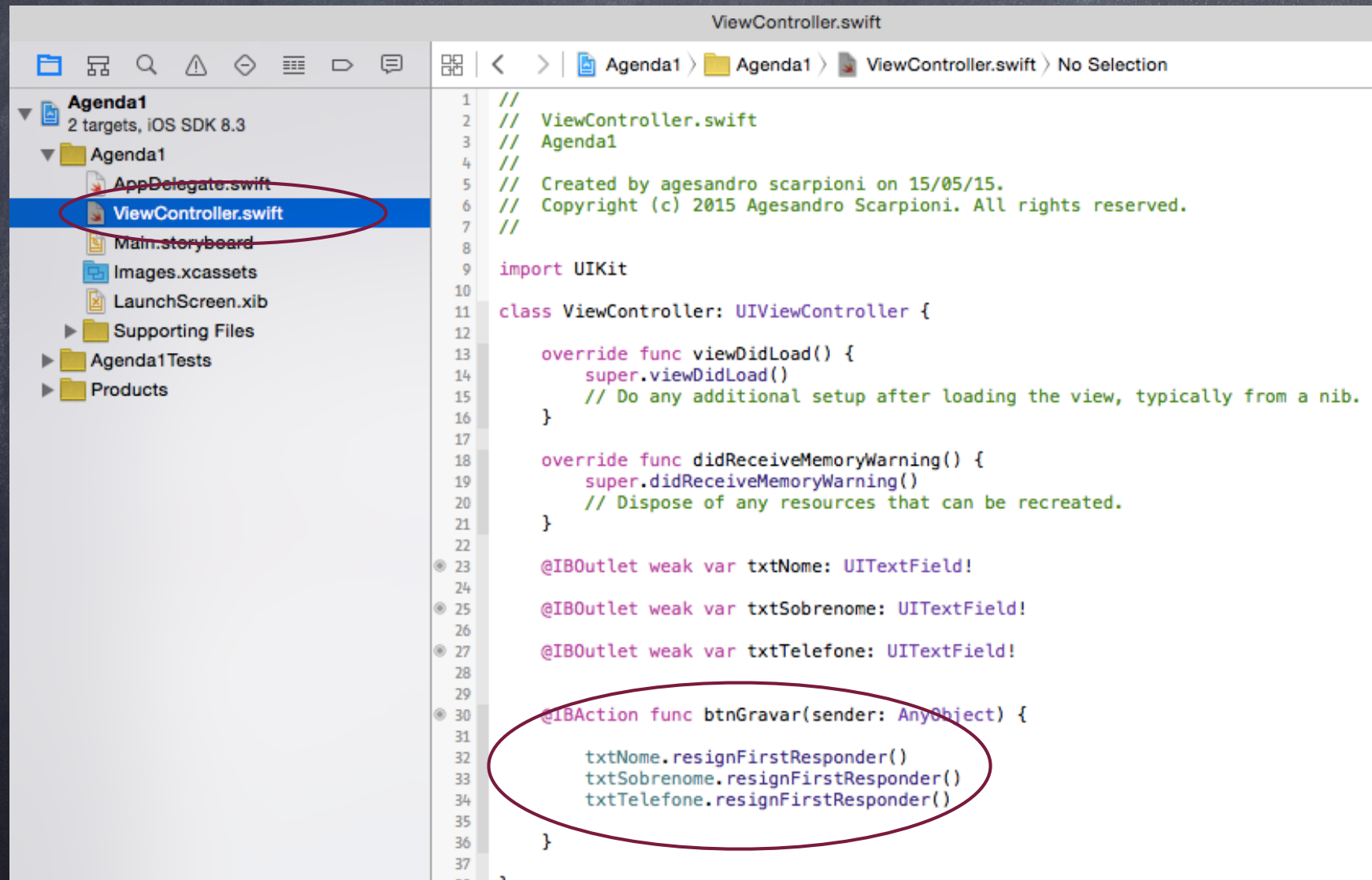
- Clique em Run ou Command + R, observe que mesmo após preencher todos os campos e depois clicando no botão gravar o teclado permanece ativo.



- Para resolver isto é necessário liberar o foco das caixas de texto, chamando o método `resignFirstResponder`, é possível chamar apenas para a caixa de telefone que é a última a ser preenchida, porém, como o usuário pode preencher em outra ordem vamos chamar o método para as três caixas.

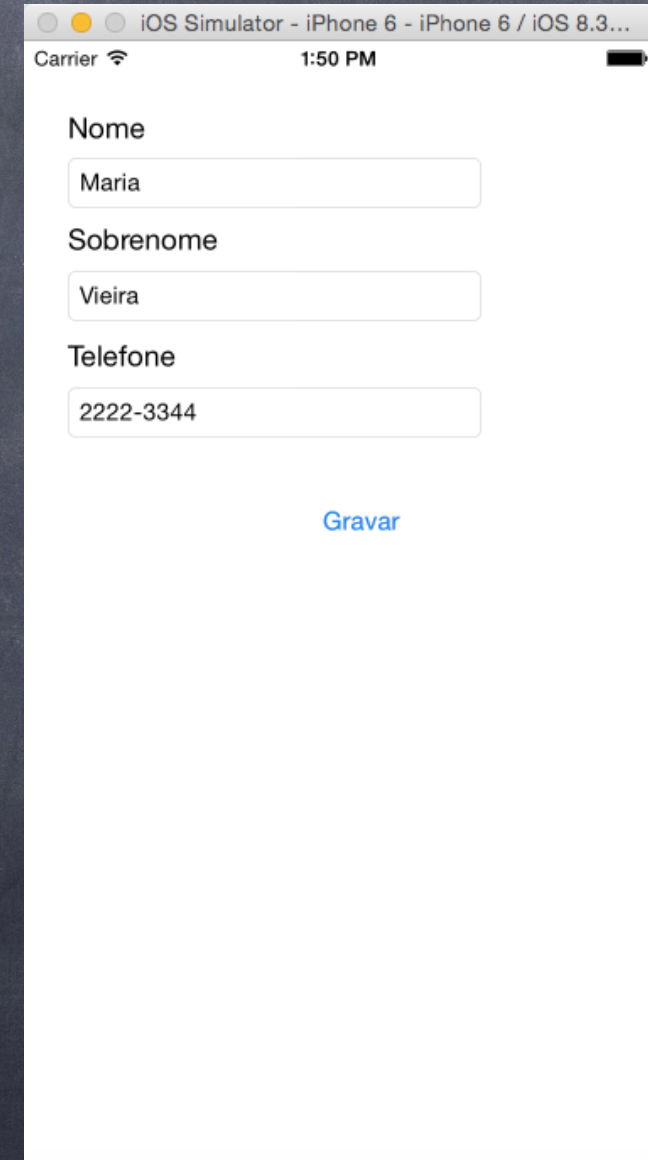
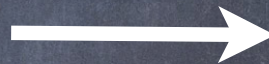
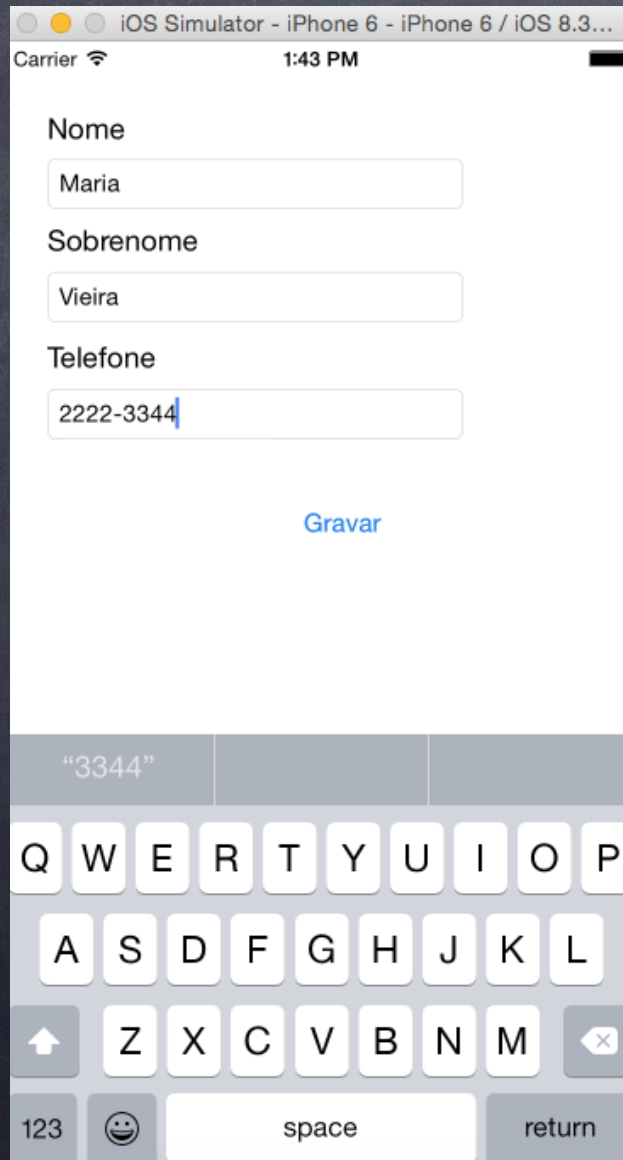
First Responder

- Faça a programação abaixo na classe viewController.swift para o IBAction do Gravar, liberando o foco das 3 caixas de texto, para isto chame o método `resignFirstResponder`.



Observe

- Clique em Run ou Command + R, observe que após o preenchimento de todos os campos e ao clicar no botão gravar o teclado desaparece.



Mais sobre o First Responder

- O First Responder é a classe que contém o foco atual da aplicação e é a primeira a responder os eventos do usuário, por isso o nome de First Responder, por Default, o First Responder é a view principal da tela onde podemos inserir outras views, quando selecionamos uma caixa de texto o foco é definido para a caixa clicada, e por este motivo o teclado aparece sendo a caixa em questão a primeira a responder os eventos do usuário. Por isso nós tivemos que liberar o foco de cada caixa após clicarmos no botão gravar para que o foco voltasse para a view e o teclado desaparecesse.

Método touchesBegan

- Uma outra possibilidade de fechar o teclado é interceptar o toque em qualquer lugar da tela utilizando o método touchesBegan da classe UIViewController, para isso crie o método no UIViewController.swift.

```

20 // Dispose of any resources that can be recreated.
21
22 Tells the responder when one or more fingers touch down in a view or window. More...
23
24 [M] touchesBegan(touches: Set<UITouch>, withEvent event: UIEvent?)
25
26 [M] touchesEstimatedPropertiesUpdated(touches: Set<NSObject>)
27
28 [M] touchesEnded(touches: Set<UITouch>, withEvent event: UIEvent?)
29
30 [M] touchesMoved(touches: Set<UITouch>, withEvent event: UIEvent?)
31
32 [M] toggleUnderline(sender: AnyObject?)
33
34 [V] topLayoutGuide: UILayoutSupport
35
36 [V] toolbarItems: [UIBarButtonItem]?
37
38 tou
39
40
41

```

- Comece digitando touc..., quando o x-Code sugerir touchesBegan dê um enter para criar o método, copie as linhas do botão Gravar.

```

28
29 @IBAction func btnGravar(sender: AnyObject) {
30     txtNome.resignFirstResponder()
31     txtSobrenome.resignFirstResponder()
32     txtTelefone.resignFirstResponder()
33 }
34
35 override func touchesBegan(touches: Set<UITouch>, withEvent event: UIEvent?) {
36     txtNome.resignFirstResponder()
37     txtSobrenome.resignFirstResponder()
38     txtTelefone.resignFirstResponder()
39 }
40
41

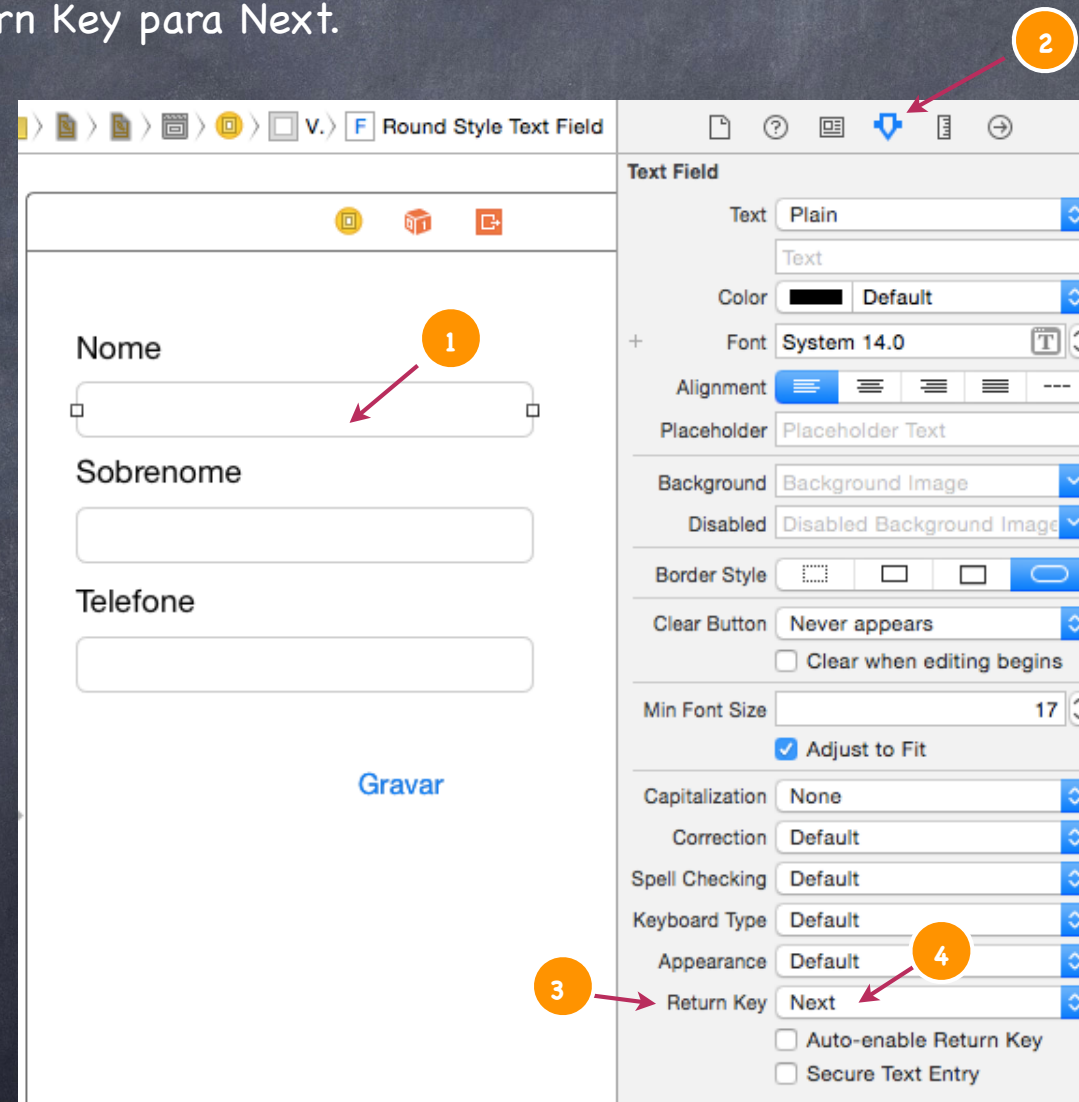
```

- Pronto, quando você clicar em qualquer lugar da tela o teclado também irá fechar.

Dica: O primeiro parâmetro do tipo NSSet do método touchesBegan contém uma lista de objetos do tipo UITouch com informações sobre o evento de touch, como por exemplo a coordenada x/y de que o mesmo ocorreu.

Modificando o teclado virtual

- É possível alterar o teclado virtual, modifique a tecla de return para next ou done, para isso selecione o campo de texto do nome, entre na guia Attributes Inspector e altere a propriedade Return Key para Next.



Obs: Faça o mesmo para o sobrenome, informando Next e para o Telefone informe Done

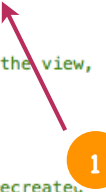
Posicionar o Foco

- Intercepte quando o usuário tocar no botão next e posicione o foco para a próxima caixa de texto, para isso será implementado um protocolo, veja os próximos slides.

Protocolo UITextFieldDelegate


- Sempre que ocorre algum evento no campo de texto, ou seja, na classe UITextField, alguns eventos são gerados, como por exemplo, o evento que ocorre quando o usuário pressiona o botão Next. Desta forma se você quiser ser notificado desse evento, a classe precisa implementar o protocolo UITextFieldDelegate, que contém justamente os métodos que se deseja interceptar.
- Um protocolo no Swift funciona de forma semelhante às interfaces do Java ou VB. No Java e no VB, seria necessário a notação implements UITextFieldDelegate, em Swift basta escrever o nome UITextFieldDelegate após uma vírgula, conforme indicado no item 1.

```
8
9  import UIKit
10
11  class ViewController: UIViewController, UITextFieldDelegate {
12
13      override func viewDidLoad() {
14          super.viewDidLoad()
15          // Do any additional setup after loading the view,
16          // typically from a nib.
17      }
18
19      override func didReceiveMemoryWarning() {
20          super.didReceiveMemoryWarning()
21          // Dispose of any resources that can be recreated.
22      }
23
24      @IBOutlet weak var txtNome: UITextField!
25      @IBOutlet weak var txtSobrenome: UITextField!
26      @IBOutlet weak var txtTelefone: UITextField!
27
28
29
30      @IBAction func btnGravar(sender: AnyObject) {
31
32          txtNome.resignFirstResponder()
33          txtSobrenome.resignFirstResponder()
34          txtTelefone.resignFirstResponder()
35
36      }
37
```



Protocolo UITextFieldDelegate

- Os métodos do protocolo UITextFieldDelegate são utilizados para que a classe receba os eventos gerados pelos campos de texto em conjunto com a utilização do teclado virtual.

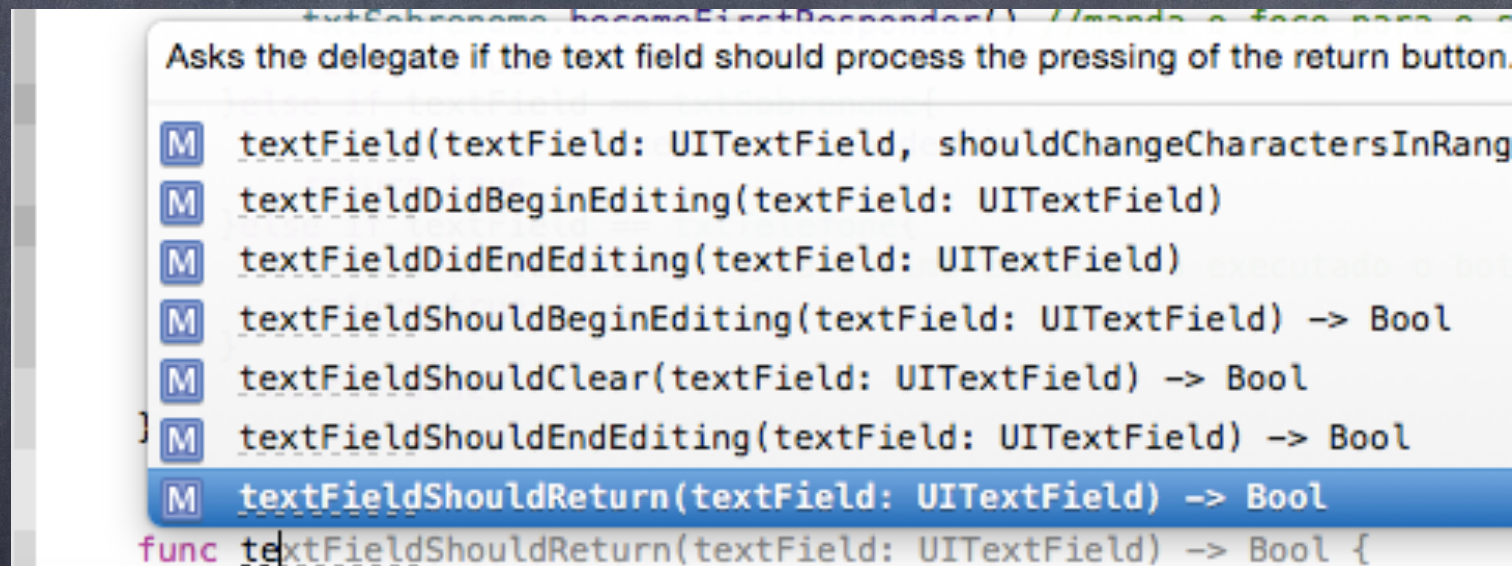


```
8
9 import UIKit
10
11 class ViewController: UIViewController, UITextFieldDelegate {
12
13     override func viewDidLoad() {
14         super.viewDidLoad()
15         // Do any additional setup after loading the view, typically from a nib.
16     }
17
18     override func didReceiveMemoryWarning() {
19         super.didReceiveMemoryWarning()
20         // Dispose of any resources that can be recreated.
21     }
22 }
```

Dica: Para implementar mais de um protocolo em uma classe separe os nomes por virgula. Ex: Protocolo1, Protocolo 2.

Protocolo UITextFieldDelegate

- Depois do método `touchesBegan`, crie um método que retorne um booleano, digite `func text...`, o xcode vai sugerir vários métodos, selecione o `textFieldShouldReturn`. Esse método é disparado quando o botão Return é pressionado.



Protocolo UITextFieldDelegate

- Digite as linha abaixo para testar se o botão da direita (Go, Next, Return, Done) do teclado virtual foi pressionado.

```
37  
38 override func touchesBegan(touches: Set<NSObject>, withEvent event: UIEvent) {  
39     txtNome.resignFirstResponder()  
40     txtSobrenome.resignFirstResponder()  
41     txtTelefone.resignFirstResponder()  
42 }  
43  
44  
45 func textFieldShouldReturn(textField: UITextField) -> Bool {  
46     if textField == txtNome{  
47         txtSobrenome.becomeFirstResponder() //manda o foco para o sobrenome  
48         return true  
49     }else if textField == txtSobrenome{  
50         txtTelefone.becomeFirstResponder() //manda o foco para o telefone  
51         return true  
52     }else if textField == txtTelefone{  
53         btnGravar(textField) //na última caixa será executado o botão gravar  
54         return true  
55     }  
56     return false  
57 }  
58
```

Obs: Isto ainda não é o suficiente para funcionar, será necessário informar ao UITextField que a classe controller.swift implementa o delegate

Protocolo UITextFieldDelegate

- Ainda no arquivo ViewController.swift, vá até o método viewDidLoad e digite as linhas para informar ao UITextField que a classe ViewController(self) implementa o delegate.

```
1 //  
2 // ViewController.swift  
3 // Agenda1  
4 //  
5 // Created by agesandro scarpioni on 15/05/15.  
6 // Copyright (c) 2015 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 import UIKit  
10  
11 class ViewController: UIViewController, UITextFieldDelegate {  
12  
13     override func viewDidLoad() {  
14         super.viewDidLoad()  
15         //Indica que a própria classe implementa o protocolo  
16         //UITextFieldDelegate para responder aos eventos  
17         txtNome.delegate = self  
18         txtSobrenome.delegate = self  
19         txtTelefone.delegate = self  
20  
21         //atenção as linhas acima podem ser substituídas por uma ligação pela tela  
22     }
```


Implementando o Protocolo UITextFieldDelegate no editor visual

- No trecho abaixo foi informado aos campos de texto quem é o delegate, ou seja, qual a classe que implementa o protocolo UITextFieldDelegate, isso foi feito via código fonte, porém, também é possível fazer a mesma coisa via editor visual. Siga os passos no próximo Slide.

```
1 //  
2 // ViewController.swift  
3 // Agenda1  
4 //  
5 // Created by agesandro scarpioni on 15/05/15.  
6 // Copyright (c) 2015 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 import UIKit  
10  
11 class ViewController: UIViewController, UITextFieldDelegate {  
12  
13     override func viewDidLoad() {  
14         super.viewDidLoad()  
15         //Indica que a própria classe implementa o protocolo  
16         //UITextFieldDelegate para responder aos eventos  
17         txtNome.delegate = self  
18         txtSobrenome.delegate = self  
19         txtTelefone.delegate = self  
20  
21         //atenção as linhas acima podem ser substituídas por uma ligação pela tela  
22     }
```


Implementando o Protocolo UITextFieldDelegate no editor visual

- Outra opção é não digitar as linhas do slide anterior (1) e informar que a classe UIViewController implementa o delegate via interface (2,3,4,5).

The image shows a screenshot of Xcode with two panels. The left panel displays the Swift code for `ViewController.swift`, and the right panel shows the Interface Builder (Storyboard) for `Main.storyboard`.

Swift Code (Left Panel):

```

1 //
2 // ViewController.swift
3 // Agenda1
4 //
5 // Created by agesandro scarpioni on 15/05/15.
6 // Copyright (c) 2015 Agesandro Scarpioni. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController, UITextFieldDelegate {
12
13     override func viewDidLoad() {
14         super.viewDidLoad()
15         //Indica que a própria classe implementa o protocolo
16         //UITextFieldDelegate para responder aos eventos
17         txtNome.delegate = self
18         txtSobrenome.delegate = self
19         txtTelefone.delegate = self
20
21         //atenção as linhas acima podem ser substituídas por uma ligação p
22     }
23 }
  
```

Interface Builder (Right Panel):

The storyboard shows a `View Controller` with three text input fields labeled `Nome`, `Sobrenome`, and `Telefone`. A `Gravar` button is at the bottom. The right sidebar shows the `Outlets` section with a `delegate` outlet. A blue line connects the `delegate` outlet to the `UITextFieldDelegate` icon in the `View Controller` class box. The `Sent Events` list includes `Did End On Exit`, `Editing Changed`, `Editing Did Begin`, `Editing Did End`, `Touch Cancel`, `Touch Down`, `Touch Down Repeat`, `Touch Drag Enter`, `Touch Drag Exit`, `Touch Drag Inside`, `Touch Drag Outside`, `Touch Up Inside`, `Touch Up Outside`, and `Value Changed`. The `Referencing Outlets` section shows `txtNome` connected to `View Controller`.

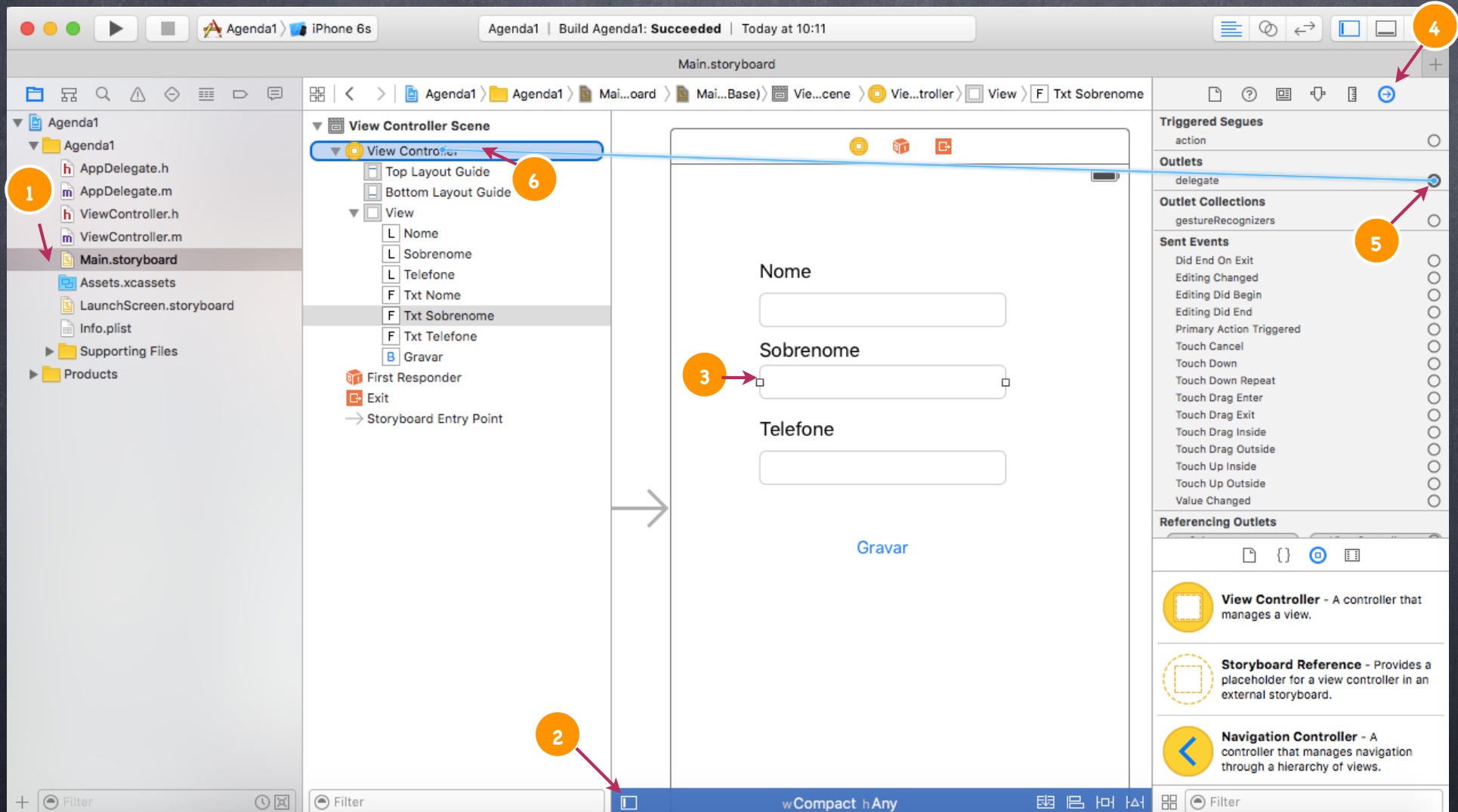
Annotations:

- 1: Points to the `UITextFieldDelegate` in the Swift code.
- 2: Points to the `UITextFieldDelegate` icon in the `View Controller` class box.
- 3: Points to the `delegate` outlet in the right sidebar.
- 4: Points to the `UITextFieldDelegate` icon in the `View Controller` class box.
- 5: Points to the `UITextFieldDelegate` icon in the `View Controller` class box.

- Repita os passos 2, 3, 4 e 5, para cada caixa de texto.

Implementando o Protocolo UITextFieldDelegate no editor visual

- Esta é a segunda forma de indicar que o ViewController implementa o UITextFieldDelegate



Implementando o Protocolo UITextFieldDelegate no editor visual

- Seguindo os passos no slide anterior você não precisa mais das linhas abaixo e as mesmas poderiam ficar comentadas. Comente as linhas 17, 18 e 19.

```
1 //
2 // ViewController.swift
3 // Agenda1
4 //
5 // Created by agesandro scarpioni on 15/05/15.
6 // Copyright (c) 2015 Agesandro Scarpioni. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController, UITextFieldDelegate {
12
13     override func viewDidLoad() {
14         super.viewDidLoad()
15         //Indica que a própria classe implementa o protocolo
16         //UITextFieldDelegate para responder aos eventos
17         txtNome.delegate = self
18         txtSobrenome.delegate = self
19         txtTelefone.delegate = self
20
21         //atenção as linhas acima podem ser substituídas por uma ligação pela tela
22     }
```


Mais sobre Protocolo UITextFieldDelegate

- Nós utilizamos no exemplo anterior o método `textFieldShouldReturn` que é chamado sempre que o botão Return, Go ou Next é pressionado. Os outros métodos possíveis deste protocolo são:
 - `textFieldDidEndEditing` - Chamado quando o campo não é mais First Responder, ou seja, o campo perdeu o foco.
 - `textFieldShouldEndEditing` - É chamado para indicar que a edição de um campo de texto pode ser finalizada, por default o método retorna YES, mas pode retornar NO, por exemplo, o teclado virtual somente será fechado se o campo de texto possuir um valor válido.
 - `textFieldDidBeginEditing` - É chamado logo depois que a edição no campo foi iniciada, ele se tornou o First Responder.
 - `textFieldShouldBeginEditing` - É chamado para verificar se o campo pode ser editado ou não, por default retorna YES, mas pode retornar NO para indicar que o campo não pode ser editado e o teclado virtual não apareça.

Testar preenchimento

- É possível fazer um teste(1) e verificar se todos os campos estão preenchidos antes de gravar uma informação, para isso, após o if será utilizado o componente UIAlertController para exibir mensagens.

```
32 @IBAction func btnGravar(sender: AnyObject) {
33     txtNome.resignFirstResponder()
34     txtSobrenome.resignFirstResponder()
35     txtTelefone.resignFirstResponder()
36     1 if txtNome.text!.isEmpty || txtSobrenome.text!.isEmpty || txtTelefone.text!.isEmpty{
37         let alerta = UIAlertController(
38             title: "Erro",
39             message: "Preencha todos os Campos",
40             preferredStyle: .Alert)
41
42         alerta.addAction(UIAlertAction(
43             title: "OK",
44             style: .Cancel,
45             handler: nil))
46
47         presentViewController(alerta,
48                             animated: true,
49                             completion: nil)
50         return
51     }
52 }
53
54
```


Testar preenchimento

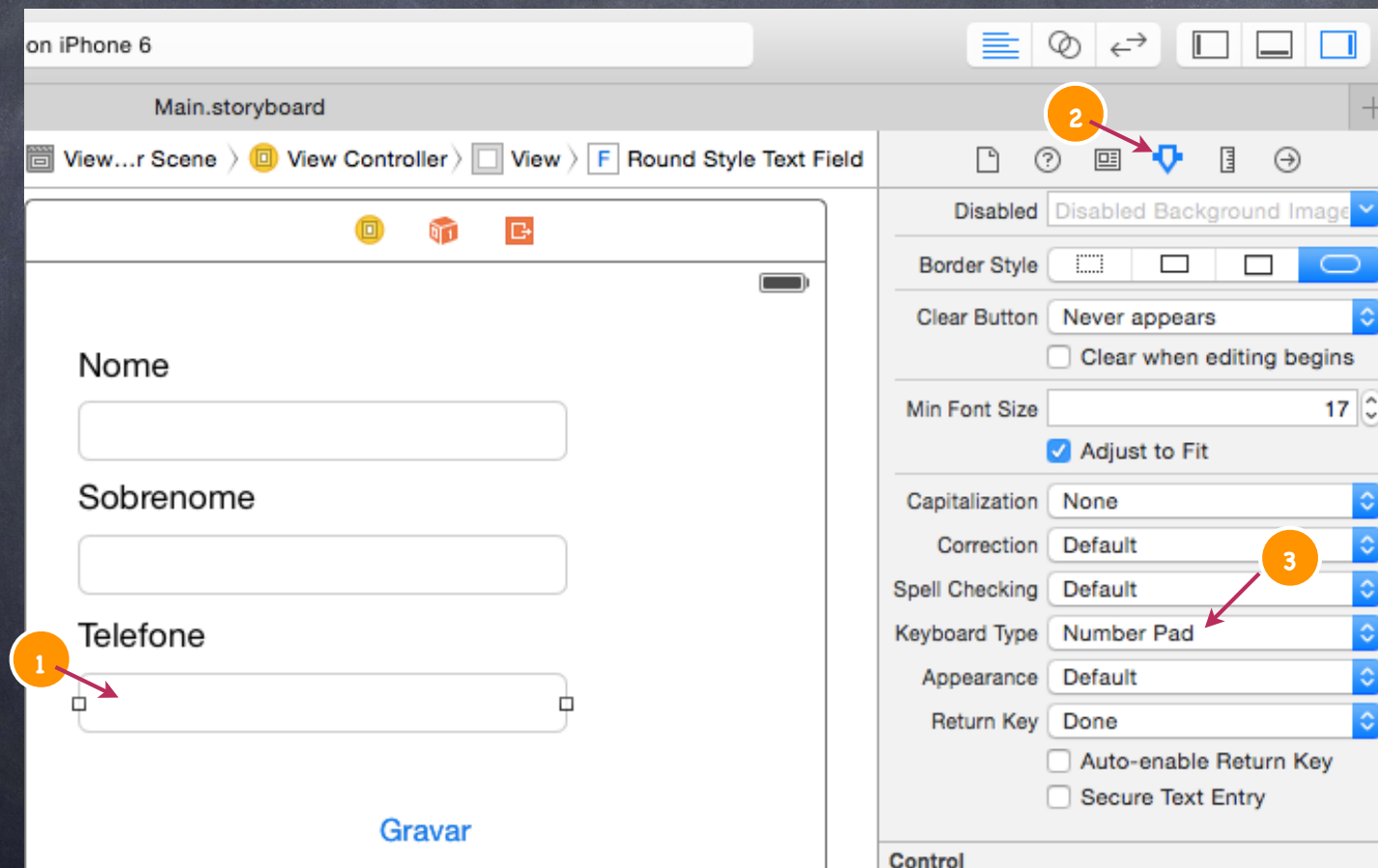
- Após o teste envie uma mensagem de dados gravados caso todos os campos tenham sido preenchidos, utilize o comando += para concatenar as caixas de texto formando uma frase qualquer e depois exibir essa frase em outra mensagem.

```
32 @IBAction func btnGravar(sender: AnyObject) {
33     txtNome.resignFirstResponder()
34     txtSobrenome.resignFirstResponder()
35     txtTelefone.resignFirstResponder()
36     if txtNome.text!.isEmpty || txtSobrenome.text!.isEmpty || txtTelefone.text!.isEmpty {
37         let alerta = UIAlertController(
38             title: "Erro",
39             message: "Preencha todos os Campos",
40             preferredStyle: .Alert)
41
42         alerta.addAction(UIAlertAction(
43             title: "OK",
44             style: .Cancel,
45             handler: nil))
46
47         presentViewController(alerta,
48                             animated: true,
49                             completion: nil)
50     }
51     return
52
53     var msg:String
54     msg = "Ok, agenda gravada para "
55     msg += txtNome.text! + " "
56     msg += txtSobrenome.text! + ""
57     msg += txtTelefone.text!
58
59     let alerta2 = UIAlertController (
60         title: "Aviso",
61         message: msg,
62         preferredStyle: UIAlertControllerStyle.Alert)
63
64     alerta2.addAction(UIAlertAction(
65         title: "OK",
66         style: UIAlertActionStyle.Cancel,
67         handler: nil))
68
69     presentViewController(alerta2,
70                         animated: true,
71                         completion: nil)
72 }
73
74 }
```

```
48                                     animated: true,
49                                     completion: nil)
50
51     }
52
53     var msg:String
54     msg = "Ok, agenda gravada para "
55     msg += txtNome.text! + " "
56     msg += txtSobrenome.text! + ""
57     msg += txtTelefone.text!
58
59     let alerta2 = UIAlertController (
60         title: "Aviso",
61         message: msg,
62         preferredStyle: UIAlertControllerStyle.Alert)
63
64     alerta2.addAction(UIAlertAction(
65         title: "OK",
66         style: UIAlertActionStyle.Cancel,
67         handler: nil))
68
69     presentViewController(alerta2,
70                         animated: true,
71                         completion: nil)
72
73 }
74 }
```

Obs: Os dados ainda não estão sendo gravados é apenas uma mensagem.

Trocar o tipo do teclado



Prática

Criação de um programa para testarmos todos os conceitos deste tópico.

- Crie um projeto novo com 4 labels (Funcionário, cargo, departamento, salário) 4 caixas de texto e 1 botão (Exibir), faça o teste para aparecer uma mensagem de erro caso todos os campos não tenham sido preenchidos, caso positivo mostre os dados concatenados em um label extra no topo da tela.
- Altere o teclado virtual com botão Next para funcionário, cargo e departamento, para salário altere o botão para Done, utilize o protocolo UITextFieldDelegate para avançar entre as caixas de texto.
- Utilize o método touchesBegan para fechar o teclado virtual quando clicarmos em qualquer parte da tela.
- Utilize o First Responder para fechar o teclado virtual quando as caixas de texto perderem o foco.