

App Agenda

parte 1 - Desenhando e preparando a Interface

X-Code - Obj-C

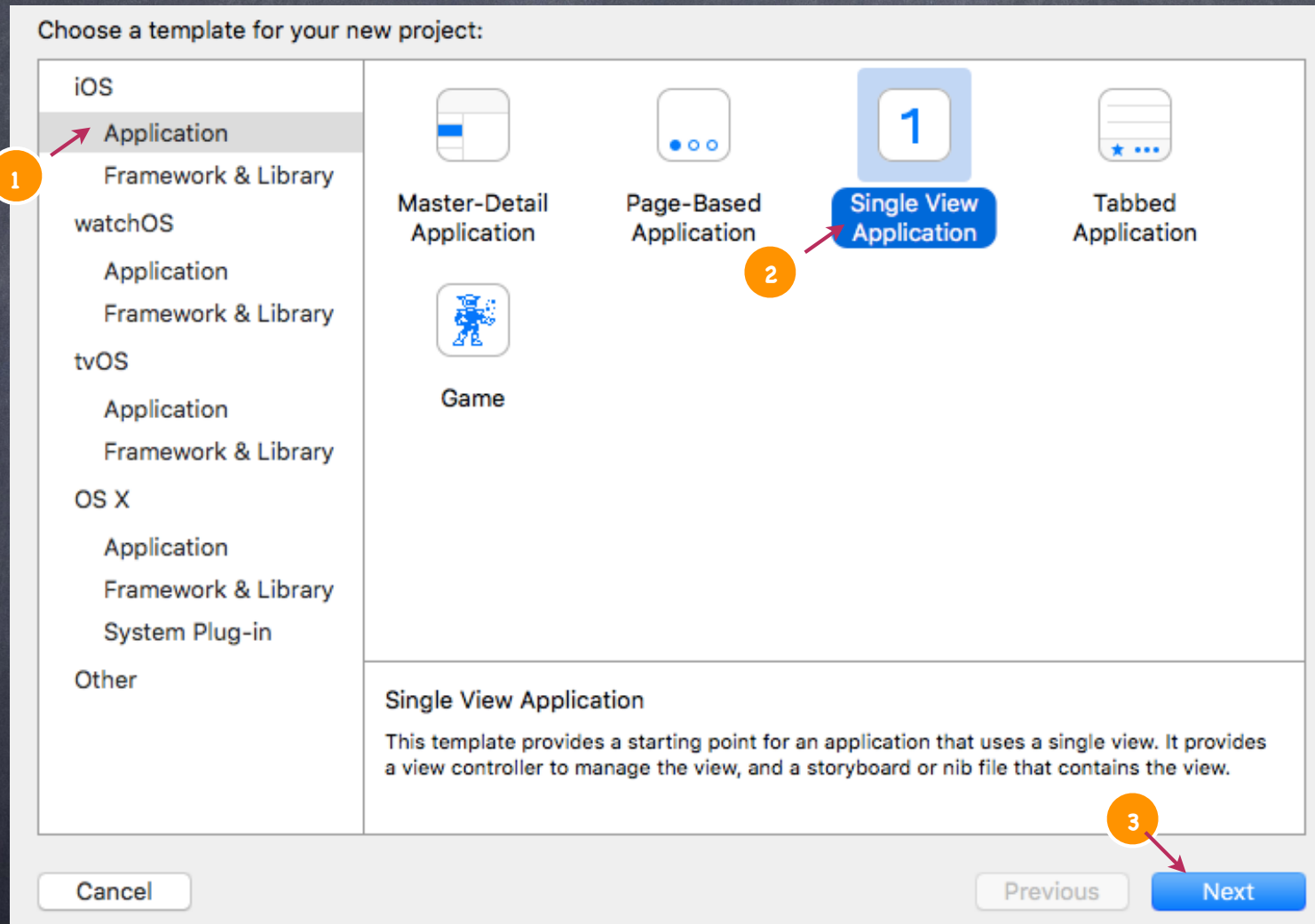
Prof. Agesandro Scarpioni

App - Agenda

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

Iniciando o Projeto

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



Caixas de Texto

- Preencha com os dados abaixo, lembre-se que o Organization Identifier funciona como se fosse o pacote no Java ou o namespace do VB, em Devices selecione iPhone e na linguagem escolha Objective-C.

Choose options for your new project:

Product Name:

Organization Name:

Organization Identifier:

Bundle Identifier:

Language:

Devices:

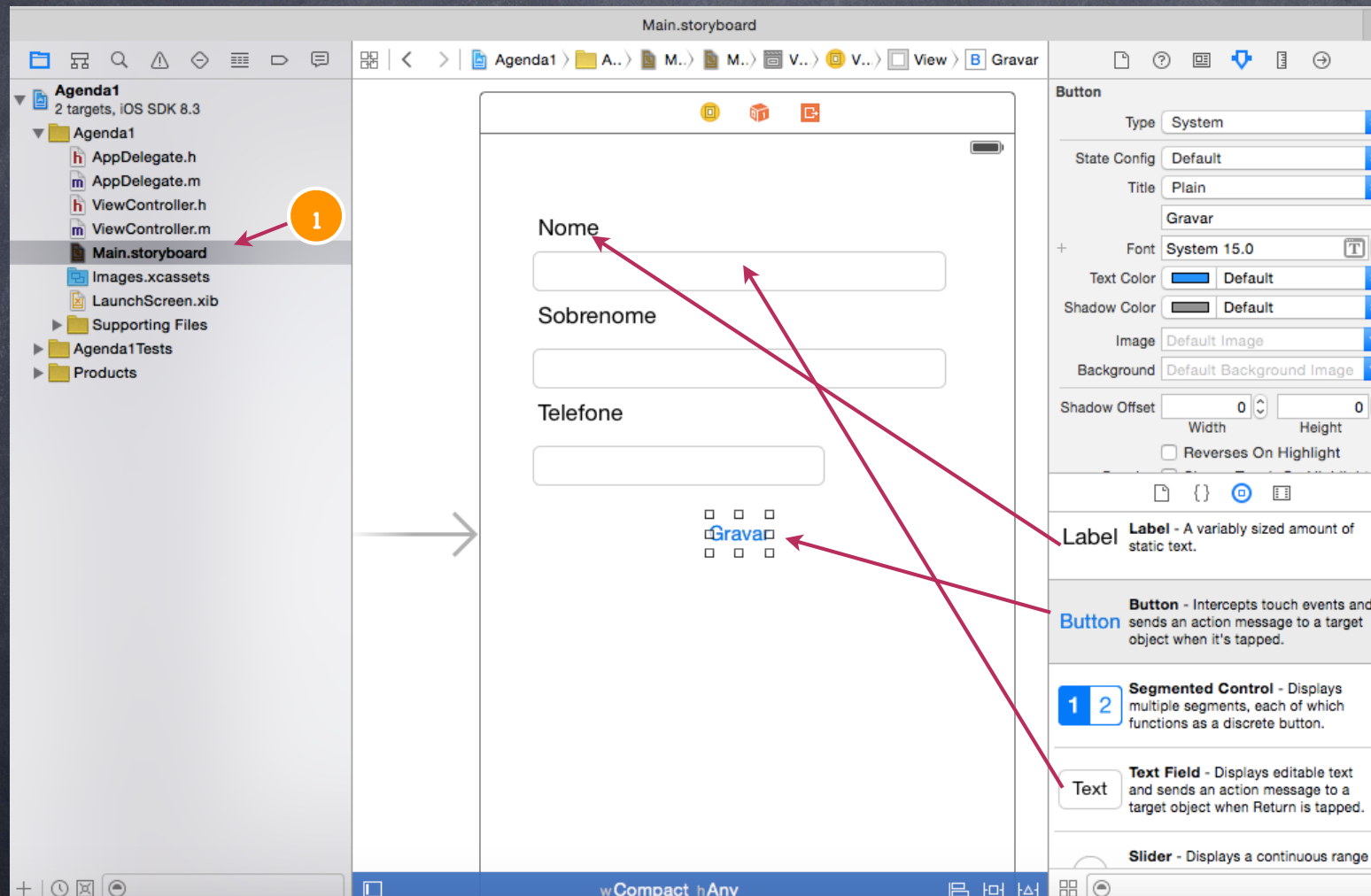
☐ Use Core Data

☐ Include Unit Tests

☐ Include UI Tests

Adicionando objetos

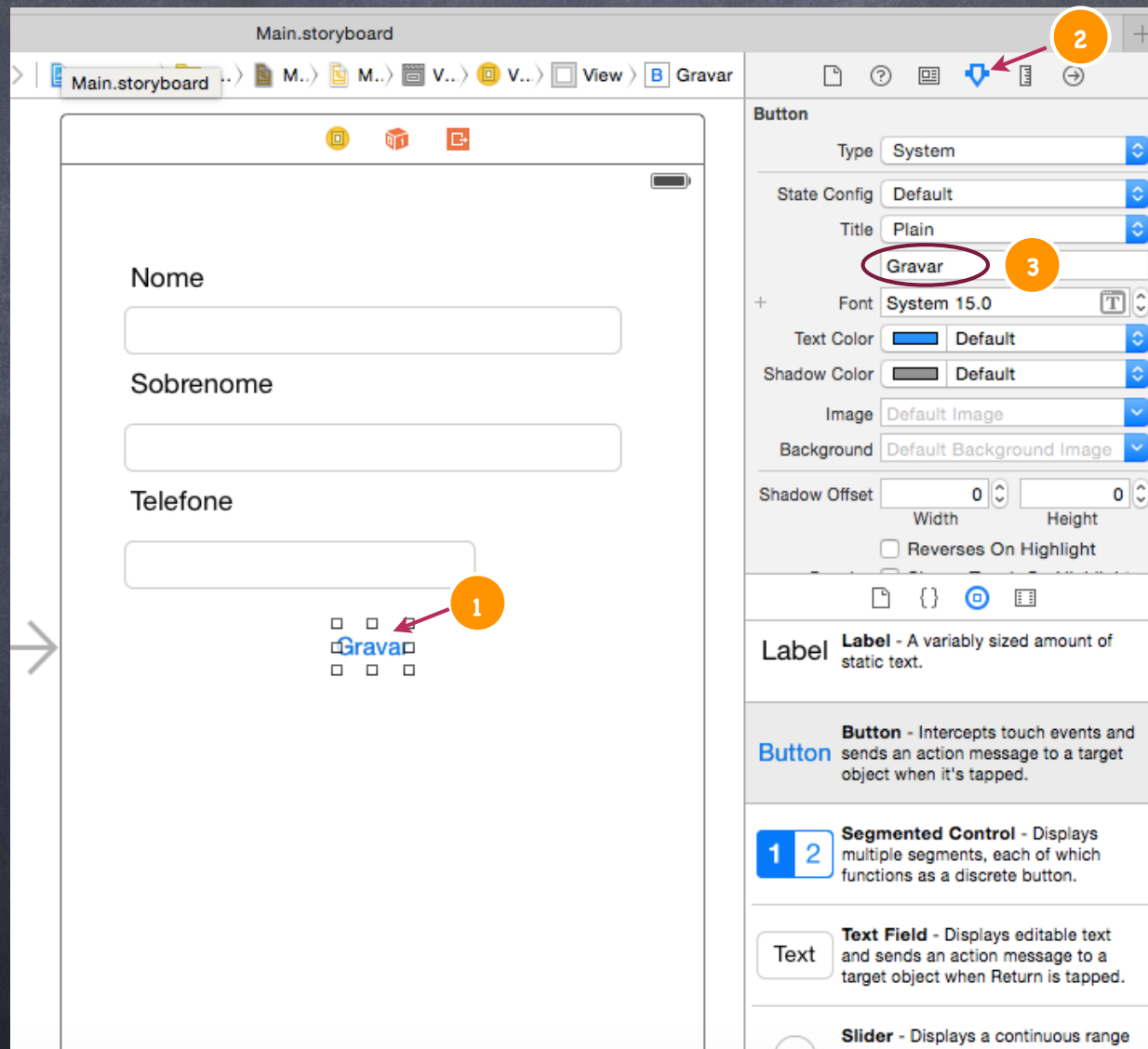
- Inclua 3 labels, 1 Button e 3 Text Field's na Storyboard do iPhone, altere o texto do botão para Gravar, e para os labels coloque Nome, Sobrenome, Telefone.



Dica: Se você esqueceu como trocar os textos dos botões e labels veja no próximo slide.

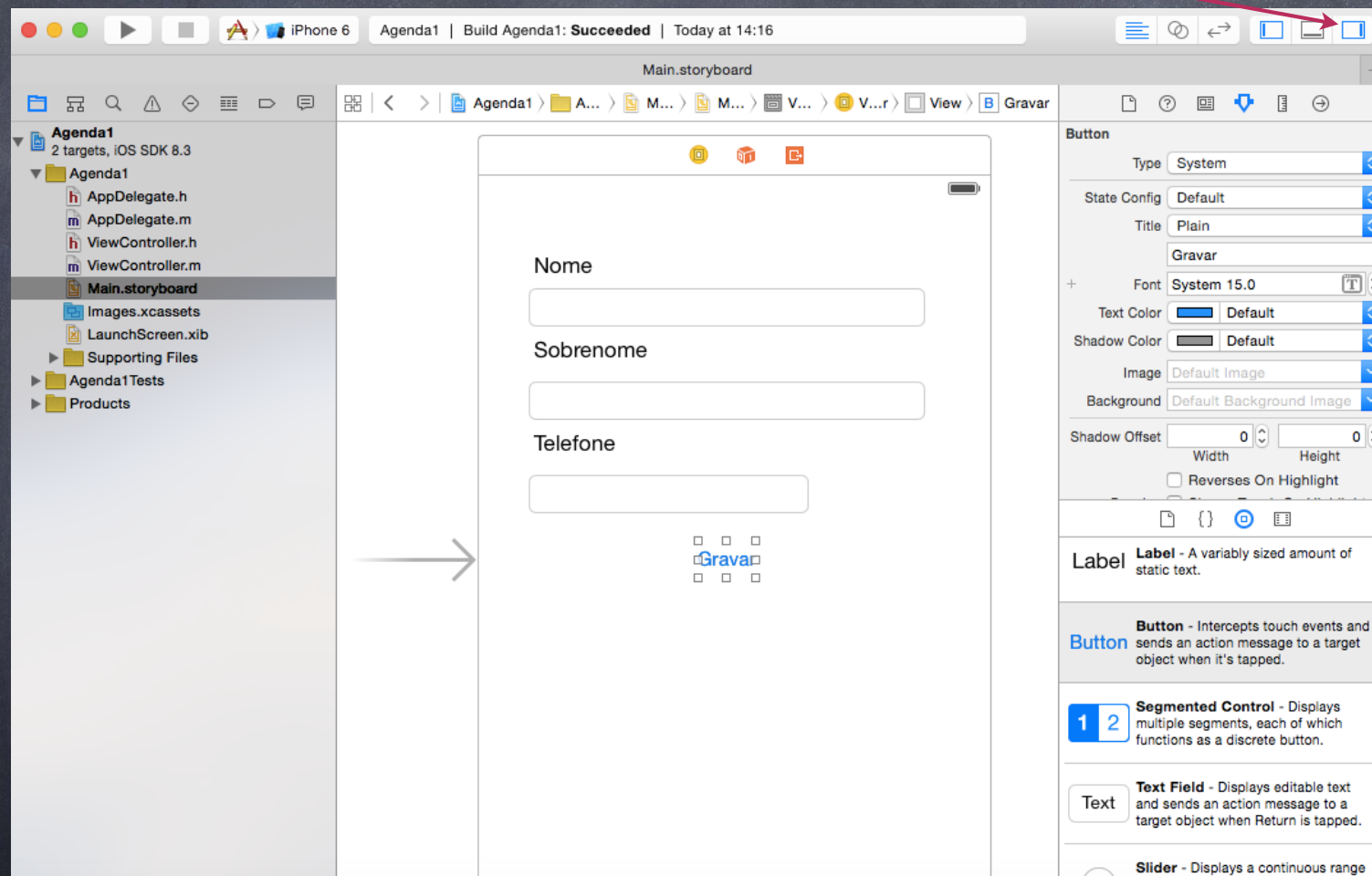
Alterando os textos

- 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 no .h

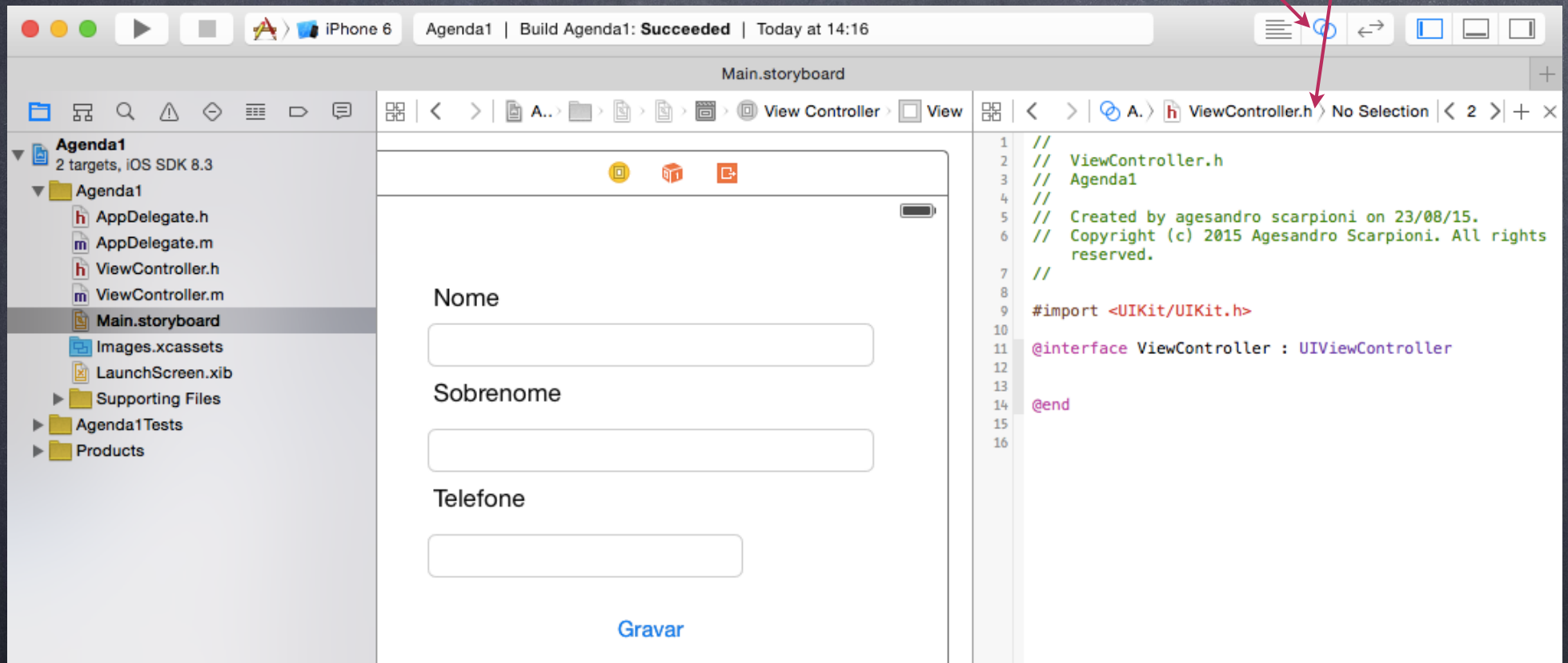
- Primeiramente organize o ambiente escondendo a janela da esquerda clicando neste ícone.



Declarando os outlet's no .h

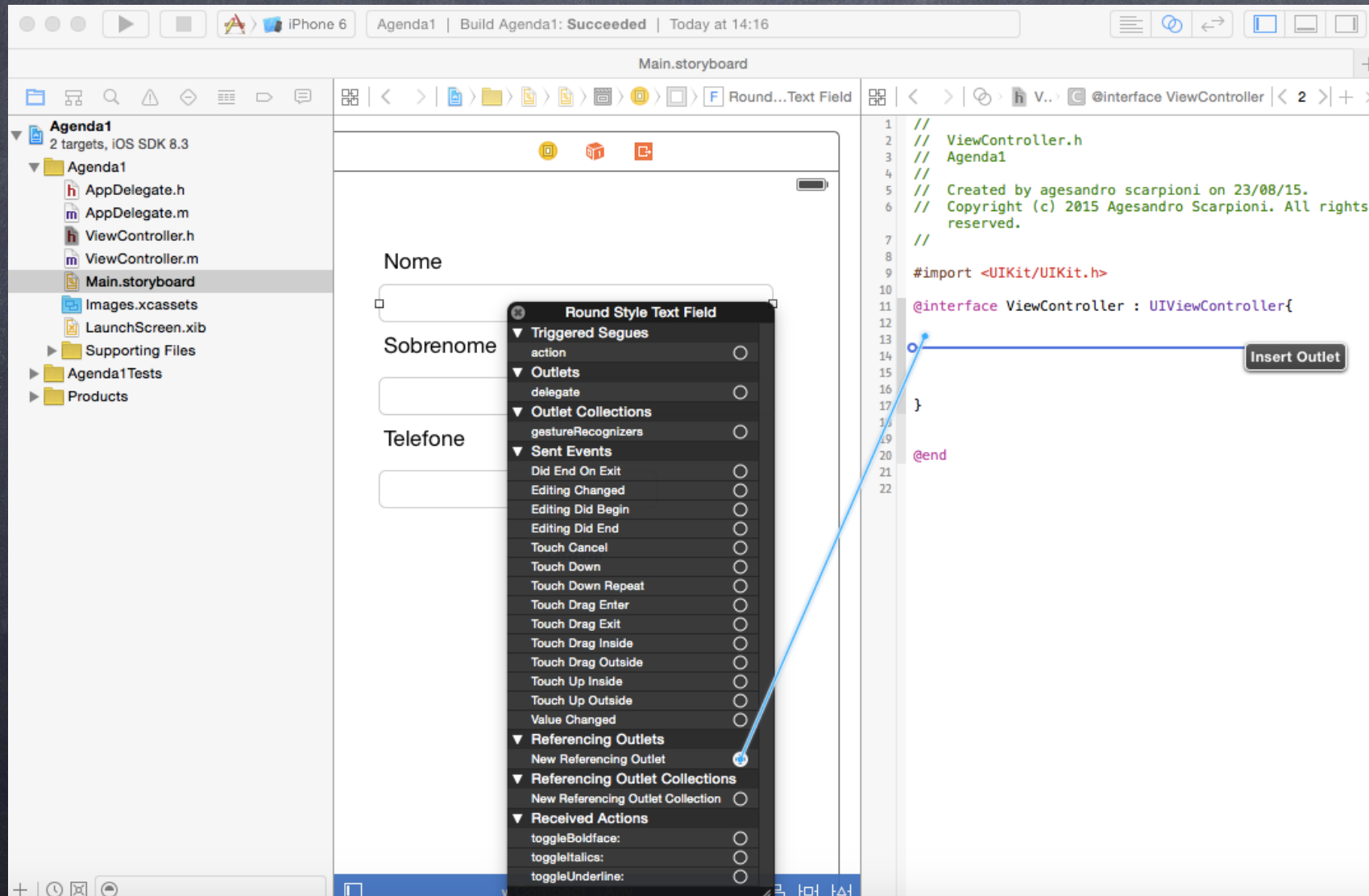
FIAP

- Depois divida a tela de Storyboard com a viewController.h obtendo as duas telas simultaneamente clicando neste ícone.



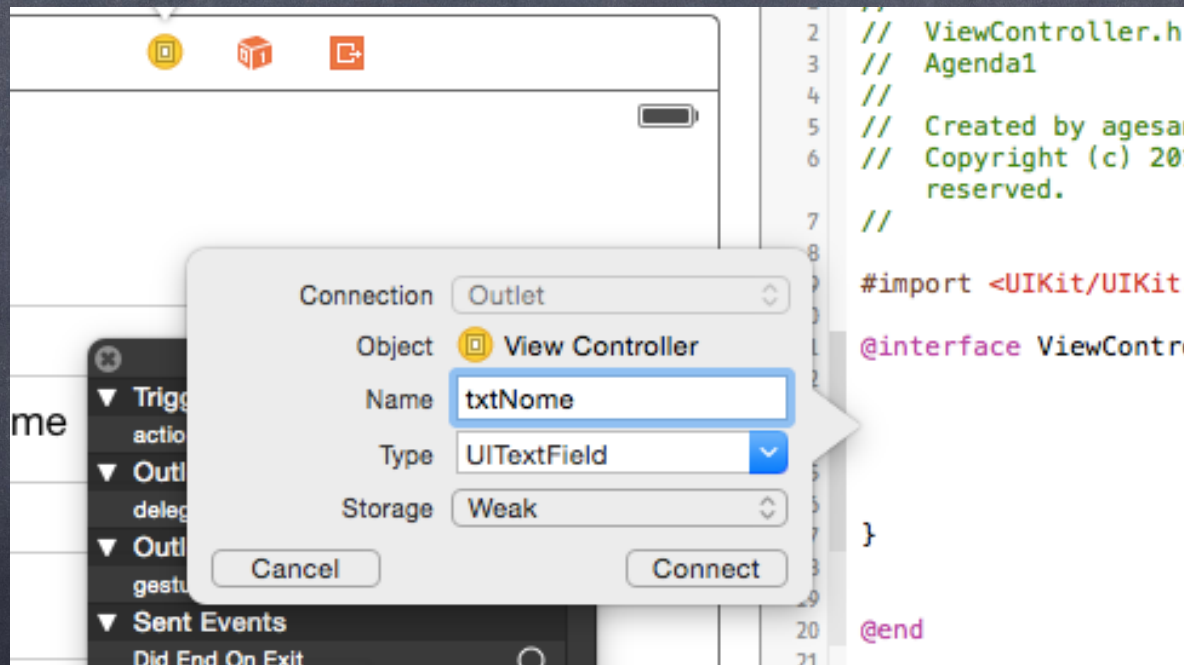
Declarando os outlet's no .h

- Coloque duas chaves { } depois da declaração da interface, 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 entre as chaves { }.



Declarando os outlet's no .h

- Quando você soltar o botão do mouse aparecerá a janela abaixo, nomeie o outlet como "txtNome" e clique em connect. Repita os mesmos passos para o 2º e 3º text's alterando os nomes respectivamente para txtSobrenome e txtTelefone.



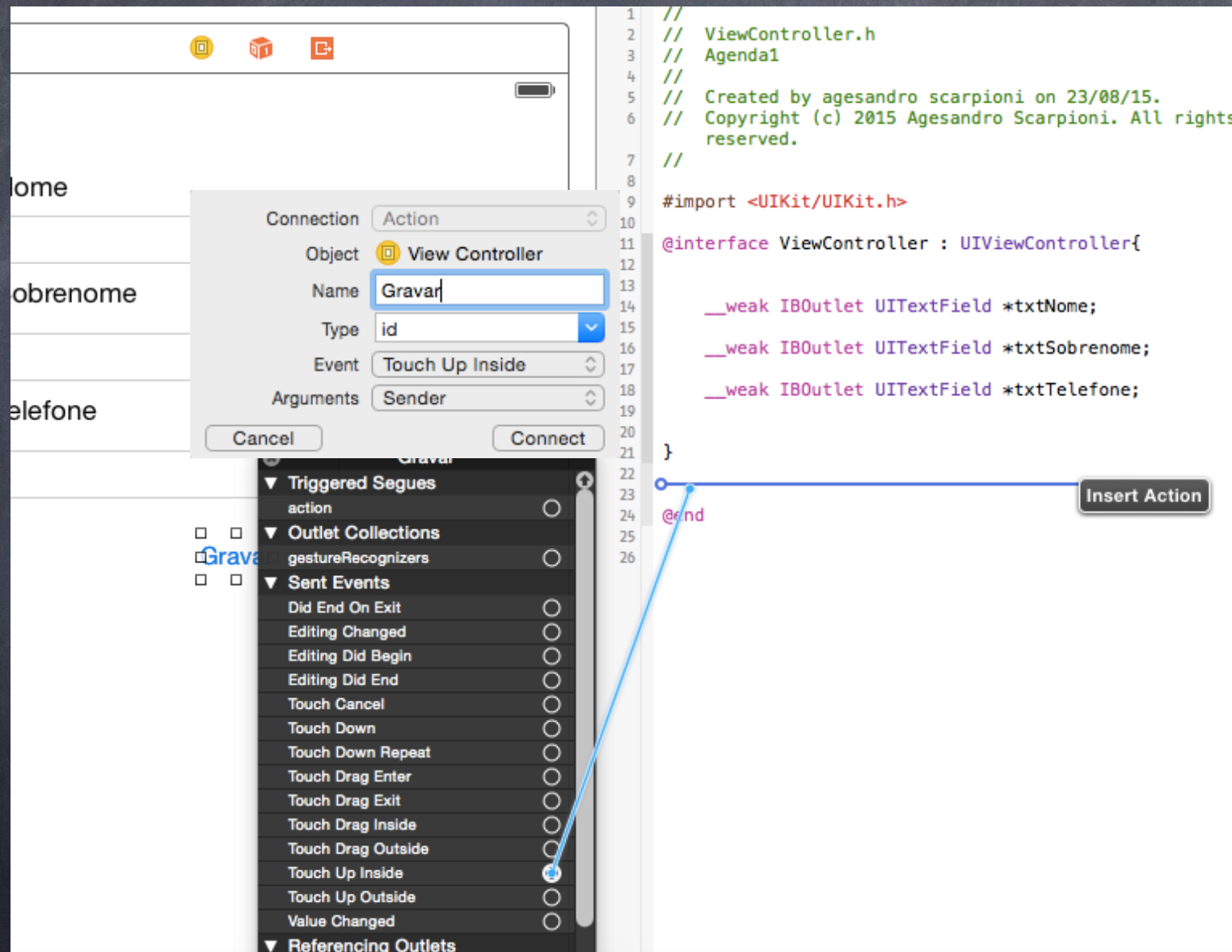
Declarando os outlet's no .h

- Ao final do processo, essas três linhas (IBOutlet's) são declaradas automaticamente no arquivo .h e os três text's já estão relacionados aos seus respectivos outlet's.

```
1 //
2 // ViewController.h
3 // Agenda1
4 //
5 // Created by agesandro scarpioni on 08/06/13.
6 // Copyright (c) 2013 Agesandro Scarpioni. All
7 // rights reserved.
8 //
9 #import <UIKit/UIKit.h>
10
11 @interface ViewController : UIViewController {
12
13
14     __weak IBOutlet UITextField *txtNome;
15
16     __weak IBOutlet UITextField *txtSobrenome;
17
18     __weak IBOutlet UITextField *txtTelefone;
19 }
20
21
22
23 @end
24
```


Declarando IBAction no .h automaticamente

- Clique com o botão direito do mouse sobre o botão Gravar, escolha o evento “Touch Up Inside” e arraste para fora da área das chaves { }.



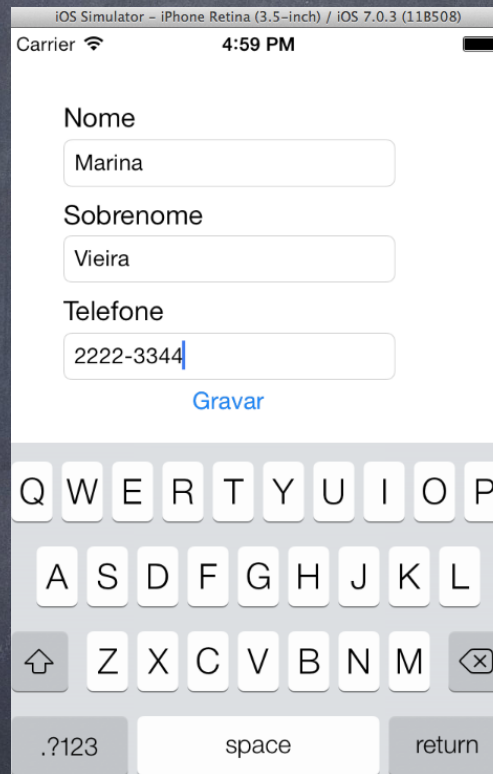
Declarando IBAction no .h automaticamente

- A declaração irá aparecer automaticamente, veja a classe ViewController.h.

```
1 //
2 // ViewController.h
3 // Agenda1
4 //
5 // Created by agesandro scarpioni on 08/06/13.
6 // Copyright (c) 2013 Agesandro Scarpioni. All
7 // rights reserved.
8
9 #import <UIKit/UIKit.h>
10
11 @interface ViewController : UIViewController {
12
13
14     __weak IBOutlet UITextField *txtNome;
15
16     __weak IBOutlet UITextField *txtSobrenome;
17
18     __weak IBOutlet UITextField *txtTelefone;
19 }
20
21 - (IBAction)Gravar:(id)sender;
22
23 @end
24
```


Execute e observe

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

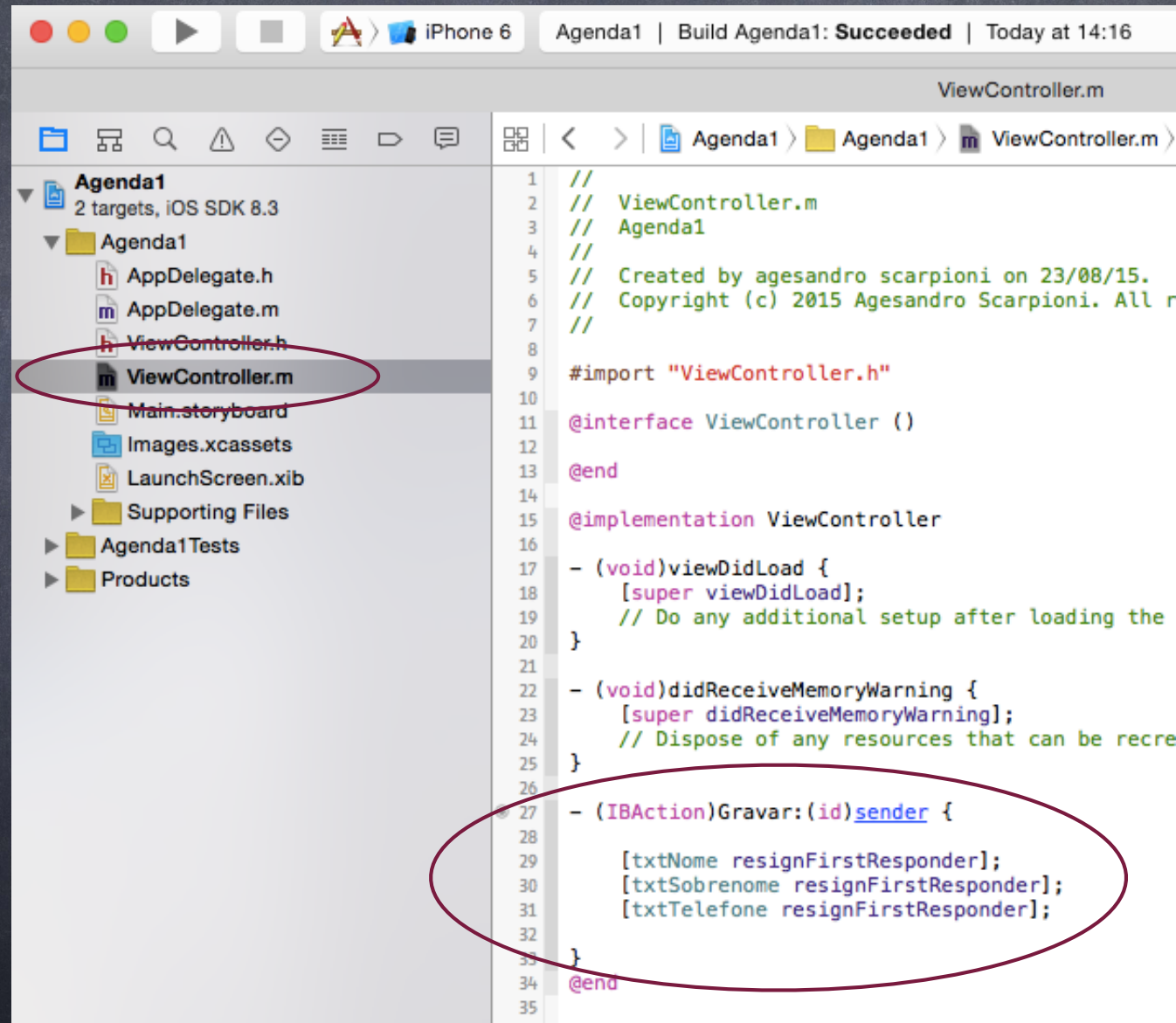


- Para resolver que o teclado seja recolhido é preciso liberar o foco das caixas de texto, chamando o método `resignFirstResponder`, é possível chamar esse método apenas para a caixa de telefone que é a última a ser preenchida, porém, como o usuário pode preencher em outra ordem, o ideal é chamar o método para as 3 caixas.

First Responder

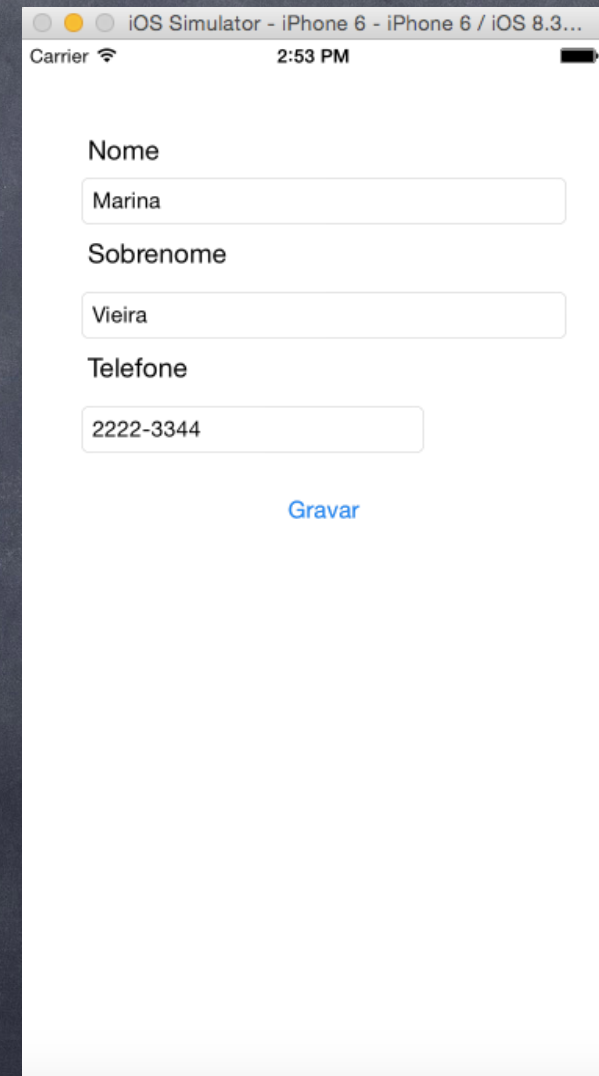
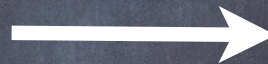
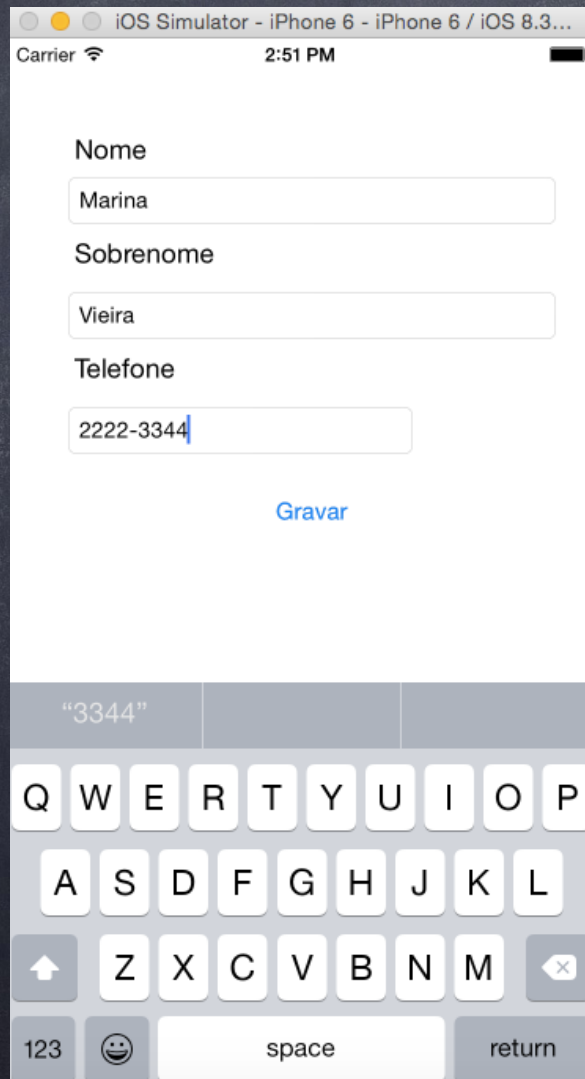
FIAP

- Faça a programação abaixo na classe viewController.m 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 ser preenchido todos os campos e depois de acionar o 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 que foi necessário liberar o foco de cada caixa após o clique 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 arquivo.m.

```

26
27 - (IBAction)Gravar:(id)sender {
28
29     [txtNome resignFirstResponder];
30     [txtSobrenome resignFirstResponder];
31     [txtTelefone resignFirstResponder];
32
33 }
34
35 -(void) touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
36
37 M touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
38 @end M touchesCancelled:(NSSet *)touches withEvent:(UIEvent *)event
39 M touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event
M touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event

```

Tells the receiver when one or more fingers touch down in a view or window.
[More...](#)

- Declare um void: - (void), comece digitando touc..., quando o X-Code sugerir touchesBegan dê um enter para criar o método, coloque as chaves e copie as linhas do botão Gravar.

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

```

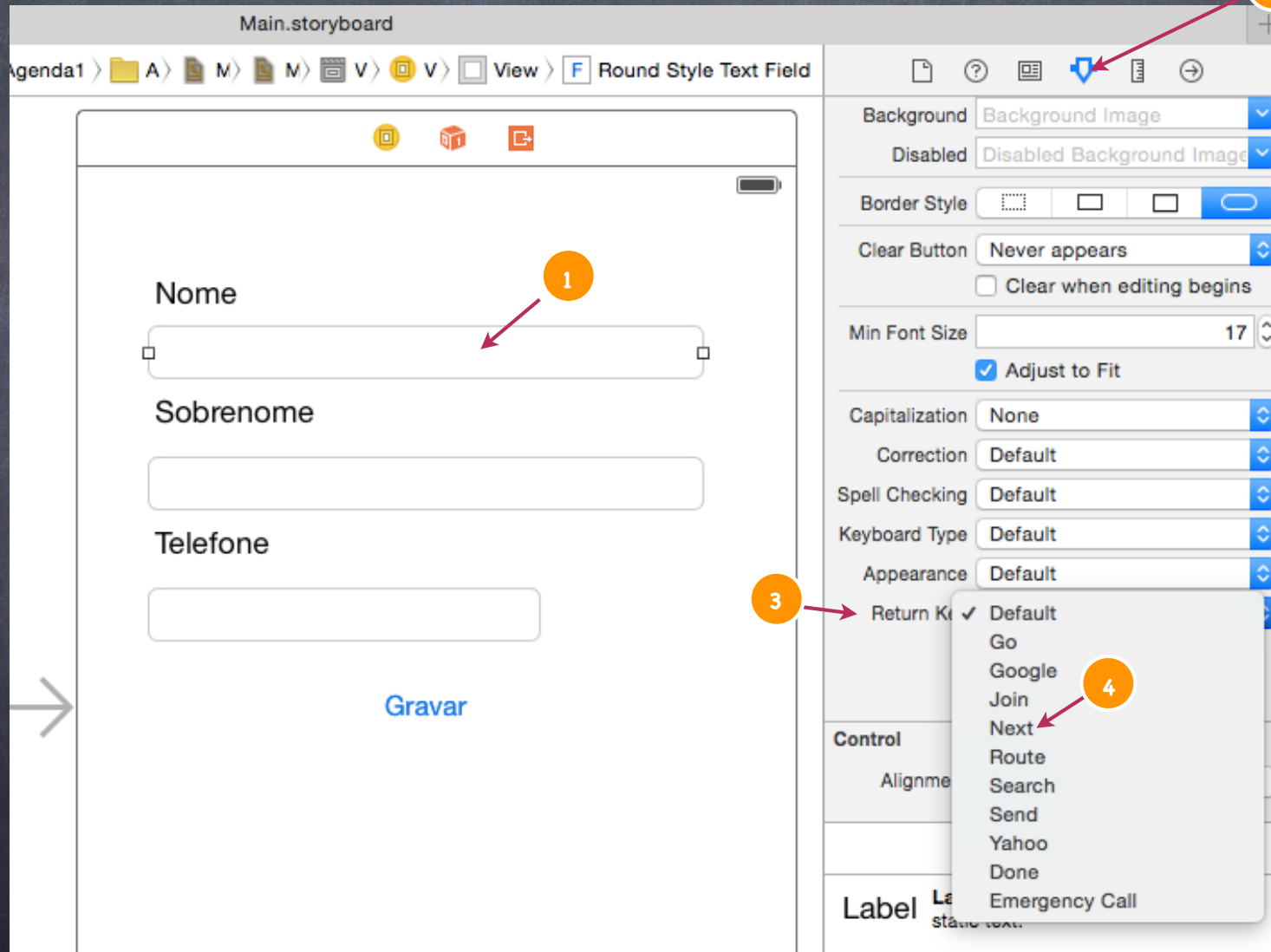
26
27 - (IBAction)Gravar:(id)sender {
28
29     [txtNome resignFirstResponder];
30     [txtSobrenome resignFirstResponder];
31     [txtTelefone resignFirstResponder];
32
33 }
34
35 -(void) touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event{
36     [txtNome resignFirstResponder];
37     [txtSobrenome resignFirstResponder];
38     [txtTelefone resignFirstResponder];
39
40 }
41

```

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 FIAP

- É possível alterar o teclado virtual modificando 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.



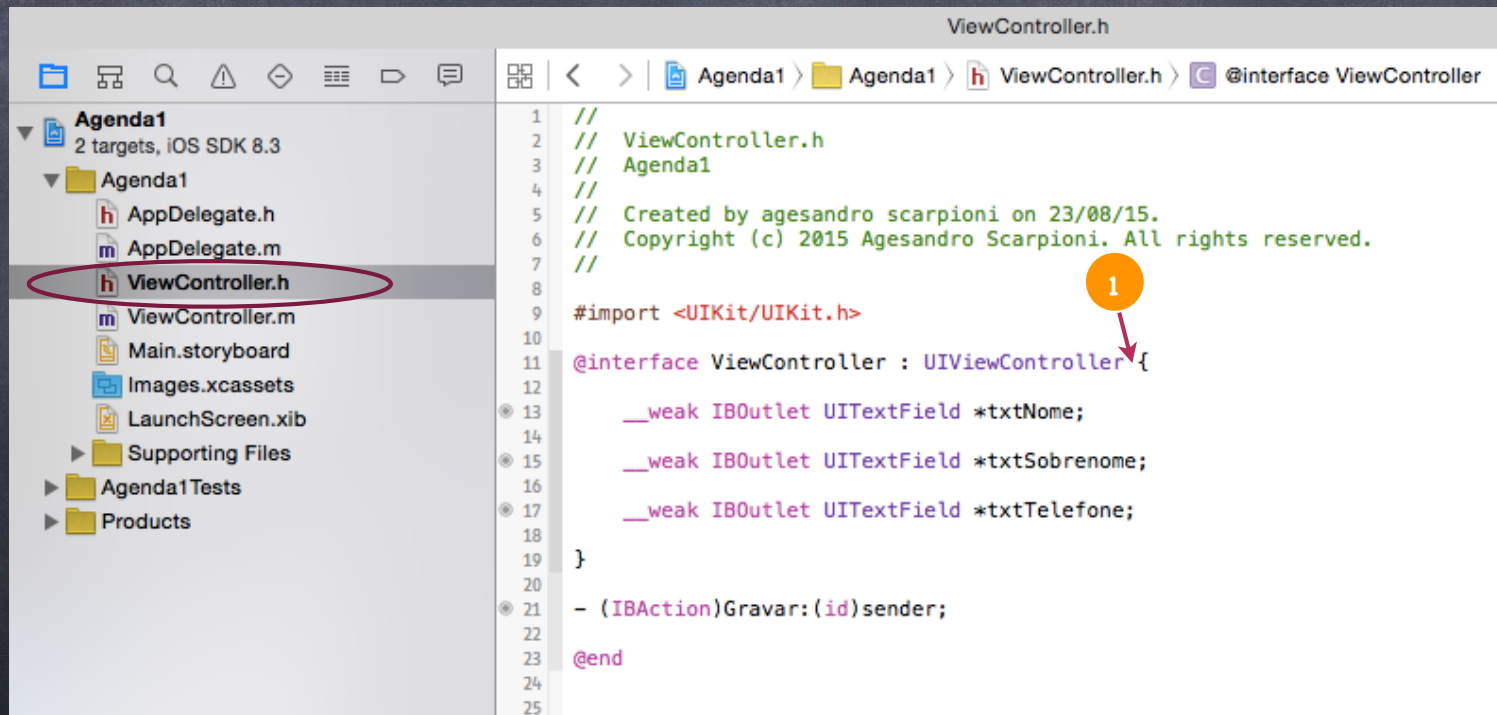
Obs: Faça o mesmo para o sobrenome, informando Next e para o Telefone informe Done

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 FIAP

- 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 quisermos ser notificados desses eventos, a classe precisa implementar o protocolo UITextFieldDelegate, que contém justamente os métodos que queremos interceptar.
- Um protocolo no Objective-C funciona de forma semelhante às interfaces do Java ou VB. No java e no VB, usaríamos a notação implements UITextFieldDelegate, em Objective-C basta escrever o nome <UITextFieldDelegate> seguidos de abre e fecha chaves.



Obs: Faça a declaração do protocolo UITextFieldDelegate no ViewController.h, como mostra o ponto 1 acima.


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.

```

1  //
2  //  ViewController.h
3  //  Agenda1
4  //
5  //  Created by agesandro scarpioni on 23/08/15.
6  //  Copyright (c) 2015 Agesandro Scarpioni. All rights reserved.
7  //
8
9  #import <UIKit/UIKit.h>
10
11  @interface ViewController : UIViewController <UITextFieldDelegate> {
12
13      __weak IBOutlet UITextField *txtNome;
14
15      __weak IBOutlet UITextField *txtSobrenome;
16
17      __weak IBOutlet UITextField *txtTelefone;
18
19  }
20
21  - (IBAction)Gravar:(id)sender;
22
23  @end
24
25

```



Dica: Para implementar mais de um protocolo em uma classe separe os nomes por virgula < __ , __ > .

UITextFieldDelegate

- Como já declaramos o protocolo no arquivo .h, agora crie o método no arquivo .m.
- Depois do método touchesBegan, crie um método que retorne um booleano, digite - (BOOL) te... , o x-Code vai sugerir vários métodos, selecione o textFieldShouldReturn, abra e feche as chaves.

```

35 -(void) touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event{
36     [txtNome resignFirstResponder];
37     [txtSobrenome resignFirstResponder];
38     [txtTelefone resignFirstResponder];
39 }
40
41
42 -(BOOL) textFieldShouldReturn:(UITextField *)textField
43 @end
44
45 [M] textField:(UITextField *)textField shouldChangeCharactersInRange:(NSRange)range replacementText:(NSString *)text
46 [M] textFieldShouldBeginEditing:(UITextField *)textField
47 [M] textFieldShouldClear:(UITextField *)textField
48 [M] textFieldShouldEndEditing:(UITextField *)textField
49 [M] textFieldShouldReturn:(UITextField *)textField
50
51 Asks the delegate if the text field should process the pressing of the return button. More...
52
53

```


UITextFieldDelegate

- Digite as linhas abaixo para testar se o botão da direita (Go, Next, Return) do teclado virtual foi pressionado. Lembre-se estas linhas são digitadas no arquivo .m.

```
38 -(void) touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event{
39     [txtNome resignFirstResponder];
40     [txtSobrenome resignFirstResponder];
41     [txtTelefone resignFirstResponder];
42
43 }
44
45 -(BOOL) textFieldShouldReturn:(UITextField *)textField{
46     if (textField == txtNome){
47         [txtSobrenome becomeFirstResponder];
48         return YES;
49     }else if (textField == txtSobrenome){
50         [txtTelefone becomeFirstResponder];
51         return YES;
52     }else if (textField == txtTelefone){
53         [self Gravar:textField];
54         return YES;
55     }
56     return NO;
57 }
58
59 @end
```

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

UITextFieldDelegate

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

```
1  //
2  //  ViewController.m
3  //  Agenda1
4  //
5  //  Created by agesandro scarpioni on 23/08/15.
6  //  Copyright (c) 2015 Agesandro Scarpioni. All rights reserved.
7  //
8
9  #import "ViewController.h"
10
11 @interface ViewController ()
12
13 @end
14
15 @implementation ViewController
16
17 - (void)viewDidLoad {
18     [super viewDidLoad];
19     //indica que a própria classe implementa o protocolo
20     //UITextFieldDelegate para responder aos eventos
21     txtNome.delegate = self;
22     txtSobrenome.delegate = self;
23     txtTelefone.delegate = self;
24 }
25
```


Implementando o Protocolo UITextFieldDelegate no editor visual

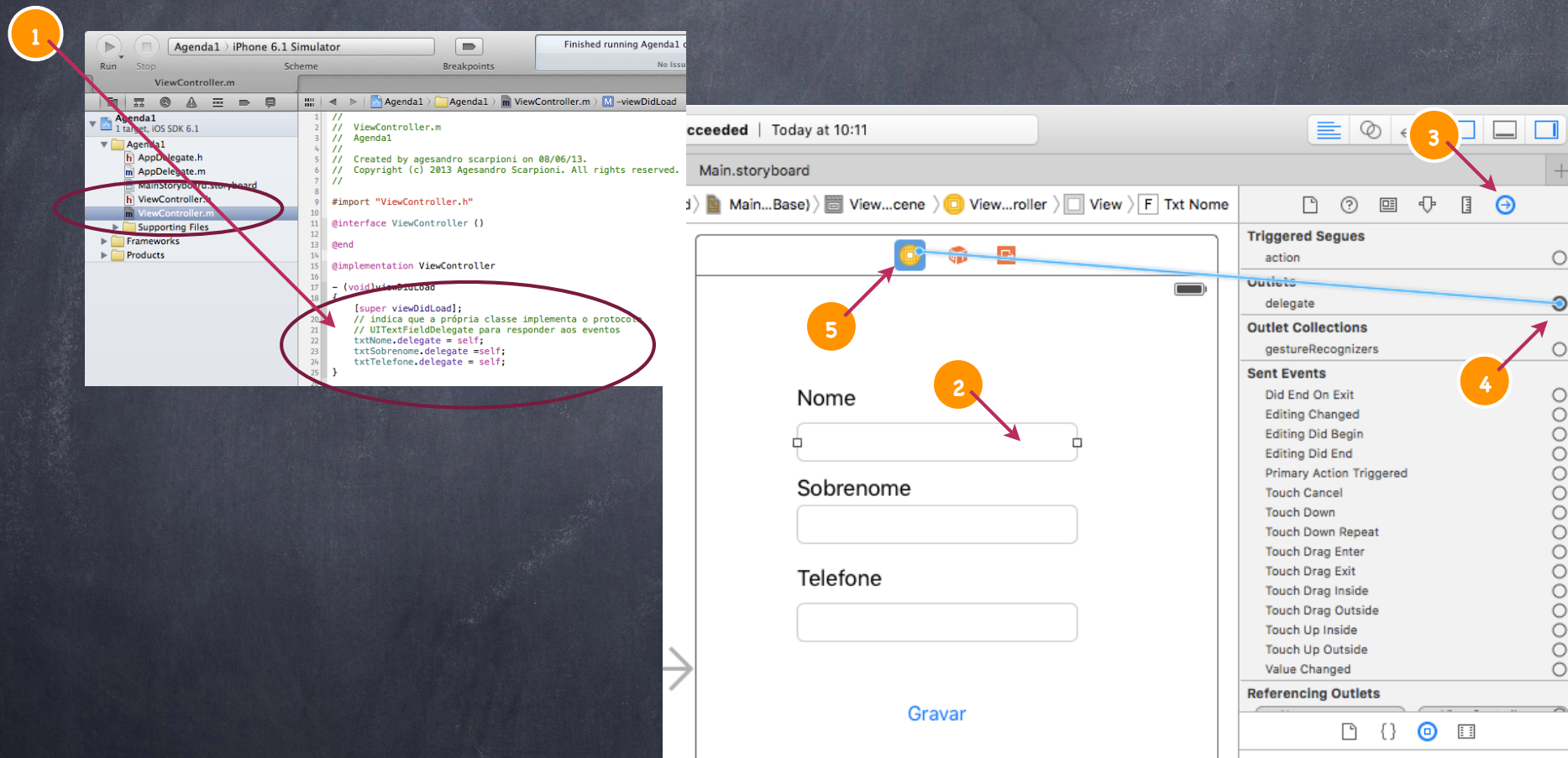
- No trecho abaixo foi informado aos campos de texto quem é o delegate, ou seja, qual a classe que implementa o protocolo UITextFieldDelegate, isso foi feito via código fonte, porém, também é possível fazer a mesma coisa via editor visual. Siga os passos no próximo Slide.

```
16
17 - (void)viewDidLoad
18 {
19     [super viewDidLoad];
20     // indica que a própria classe implementa o protocolo
21     // UITextFieldDelegate para responder aos eventos
22     txtNome.delegate = self;
23     txtSobrenome.delegate = self;
24     txtTelefone.delegate = self;
25 }
```


Implementando o Protocolo UITextFieldDelegate no editor visual

FIAP

- 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).

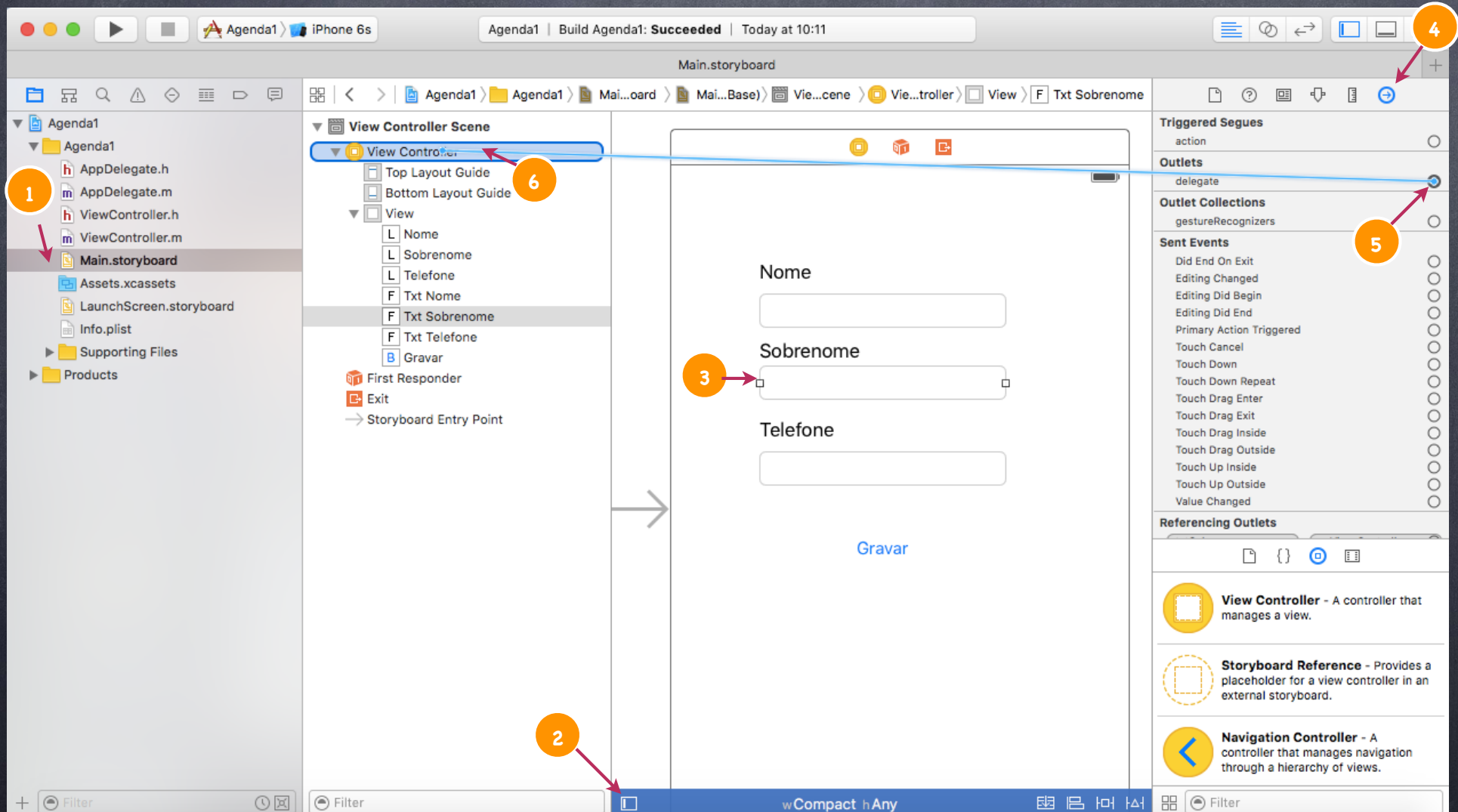


Obs: Repita o passo 3 e 4 para cada caixa de texto

Implementando o Protocolo UITextFieldDelegate no editor visual

FIAP

- 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 22, 23 e 24.

```
16
17 - (void)viewDidLoad
18 {
19     [super viewDidLoad];
20     // indica que a própria classe implementa o protocolo
21     // UITextFieldDelegate para responder aos eventos
22     txtNome.delegate = self;
23     txtSobrenome.delegate = self;
24     txtTelefone.delegate = self;
25 }
```


Mais sobre Protocolo FIAP UITextFieldDelegate

- Foi utilizado 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 FIAP

- É possível fazer um teste(1) e verificar se todos os campos estão preenchidos antes de gravar uma informação.

```
30
31 - (IBAction)txtGravar:(id)sender {
32     [txtNome resignFirstResponder];
33     [txtSobrenome resignFirstResponder];
34     [txtTelefone resignFirstResponder];
35
36     if ([txtNome.text isEqualToString:@""] || [txtSobrenome.text isEqualToString:@""] ||
37         [txtTelefone.text isEqualToString:@""]){
38
39         UIAlertController *alerta = [UIAlertController
40                                     alertControllerWithTitle:@"Aviso"
41                                     message:@"Preencha todos os campos"
42                                     preferredStyle:UIAlertControllerStyleAlert];
43
44         UIAlertAction *ok = [UIAlertAction
45                             initWithTitle:@"OK"
46                             style:UIAlertActionStyleDefault
47                             handler:^(UIAlertAction * _Nonnull action) {
48                                 [alerta dismissViewControllerAnimated:YES completion:nil];
49                             }];
50
51         [alerta addAction:ok];
52
53         [self presentViewController:alerta animated:YES completion:nil];
54         return;
55     }
56 }
```


Testar preenchimento

FIAP

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

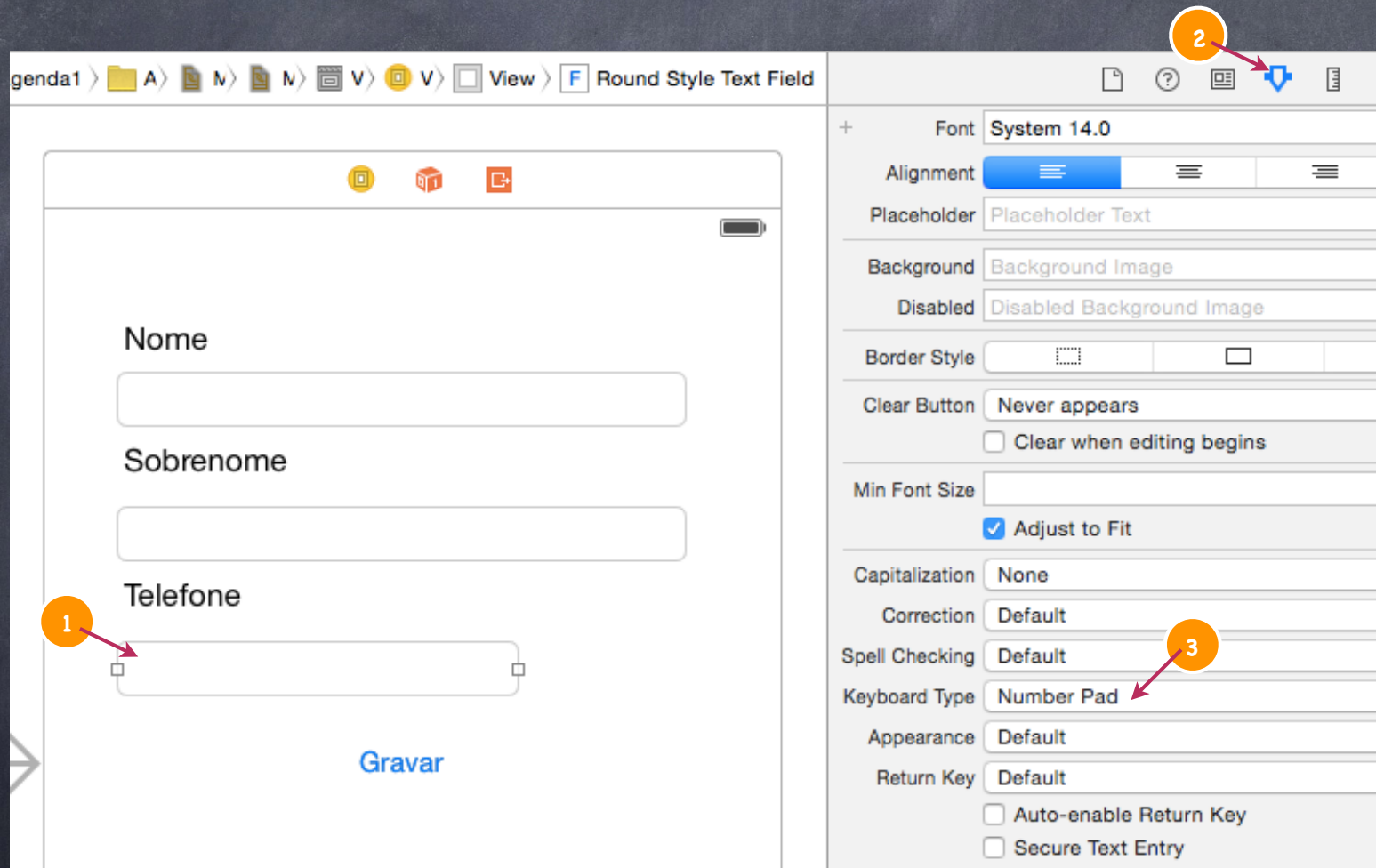
```
31 - (IBAction)txtGravar:(id)sender {
32     [txtNome resignFirstResponder];
33     [txtSobrenome resignFirstResponder];
34     [txtTelefone resignFirstResponder];
35
36     if ([txtNome.text isEqualToString:@""] || [txtSobrenome.text isEqualToString:@""] ||
37         [txtTelefone.text isEqualToString:@""]){
38
39         UIAlertController *alerta = [UIAlertController
40                                     alertControllerWithTitle:@"Aviso"
41                                     message:@"Preencha todos os campos"
42                                     preferredStyle:UIAlertControllerStyleAlert];
43
44         UIAlertAction *ok = [UIAlertAction
45                             actionWithTitle:@"OK"
46                             style:UIAlertActionStyleDefault
47                             handler:^(UIAlertAction * _Nonnull action) {
48                                 [alerta dismissViewControllerAnimated:YES completion:nil];
49                             }];
50
51         [alerta addAction:ok];
52         [self presentViewController:alerta animated:YES completion:nil];
53     }
54
55     NSString *texto = @"OK agenda gravada para ";
56     texto = [texto stringByAppendingString:txtNome.text];
57     texto = [texto stringByAppendingString:@" "];
58     texto = [texto stringByAppendingString:txtSobrenome.text];
59
60     UIAlertController *alerta2 = [UIAlertController
61                                   alertControllerWithTitle:@"Aviso"
62                                   message:texto
63                                   preferredStyle:UIAlertControllerStyleAlert];
64
65     UIAlertAction *ok2 = [UIAlertAction
66                           actionWithTitle:@"OK"
67                           style:UIAlertActionStyleDefault
68                           handler:^(UIAlertAction * _Nonnull action) {
69                               [alerta2 dismissViewControllerAnimated:YES completion:nil];
70                           }];
71
72     [alerta2 addAction:ok2];
73     [self presentViewController:alerta2 animated:YES completion:nil];
74 }
75
76 }
```

```
52     [self presentViewController:alerta animated:YES completion:nil];
53     return;
54 }
55
56 NSString *texto = @"OK agenda gravada para ";
57 texto = [texto stringByAppendingString:txtNome.text];
58 texto = [texto stringByAppendingString:@" "];
59 texto = [texto stringByAppendingString:txtSobrenome.text];
60
61 UIAlertController *alerta2 = [UIAlertController
62                               alertControllerWithTitle:@"Aviso"
63                               message:texto
64                               preferredStyle:UIAlertControllerStyleAlert];
65
66 UIAlertAction *ok2 = [UIAlertAction
67                       actionWithTitle:@"OK"
68                       style:UIAlertActionStyleDefault
69                       handler:^(UIAlertAction * _Nonnull action) {
70                           [alerta2 dismissViewControllerAnimated:YES completion:nil];
71                       }];
72
73 [alerta2 addAction:ok2];
74
75 [self presentViewController:alerta2 animated:YES completion:nil];
76 }
```

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

Trocar o tipo do teclado

- Configure o teclado para exibir apenas números para o campo telefone.



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.