

Core Data

Preparando a interface - Parte 1

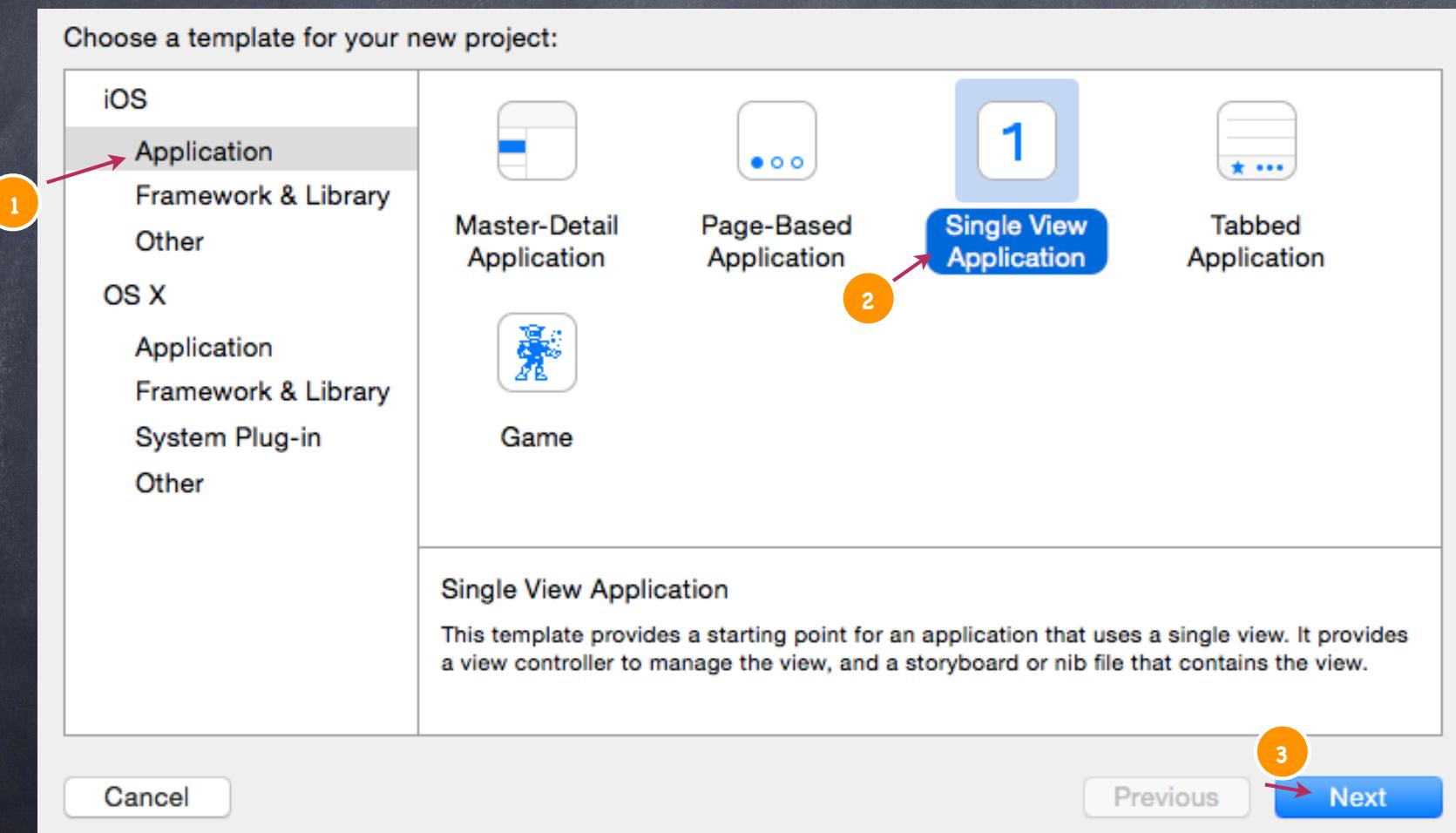
X-Code com Objective-C
Prof. Agesandro Scarpioni

Core Data

- ➊ Vamos gravar os dados e ver como é fácil navegar entre telas utilizando o storyboard, o storyboard foi implementado a partir do IOS 5 com Xcode4.

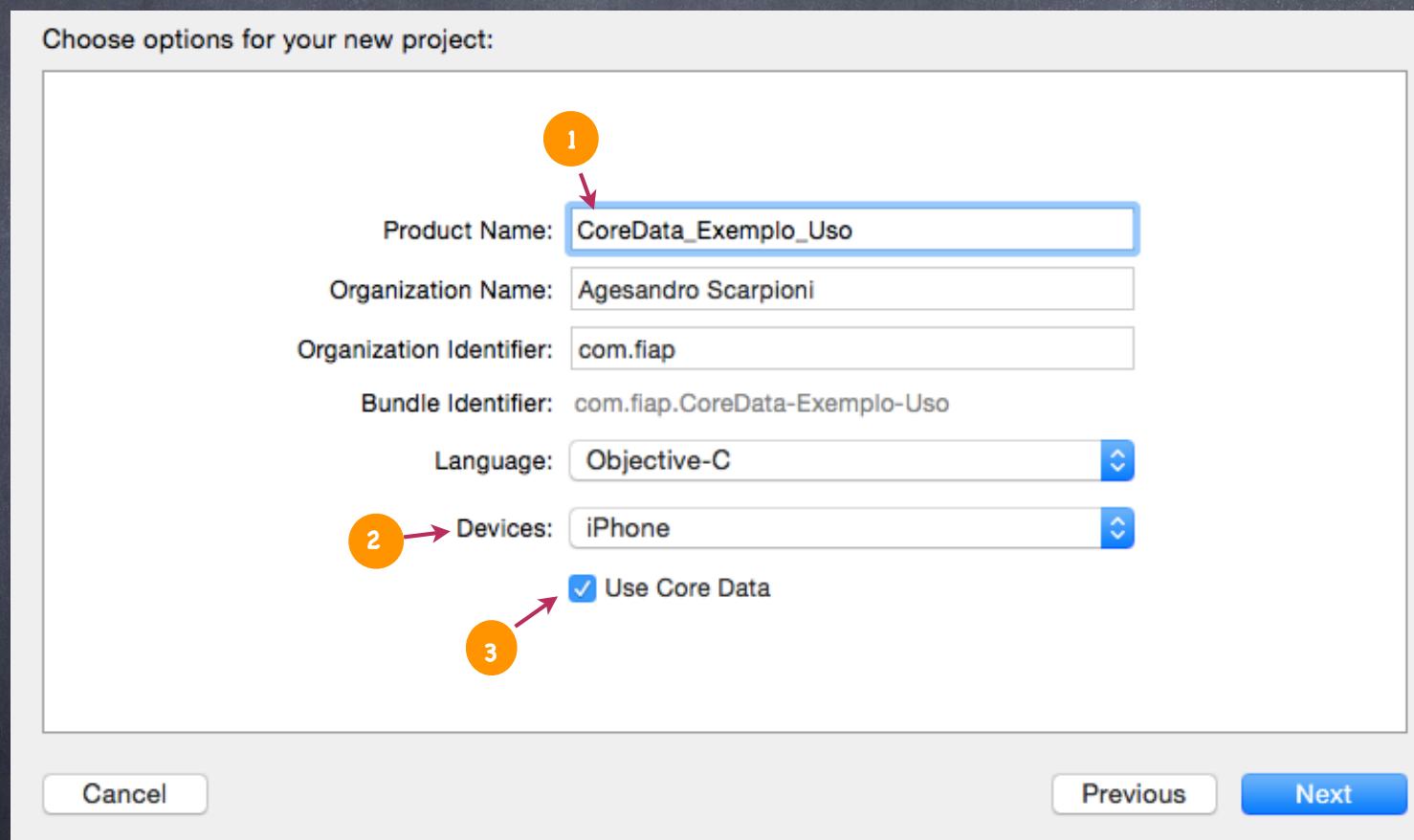
Core Data

- Vamos criar um projeto novo do tipo IOS application (1) - Single View Application (2) clique em Next(3).



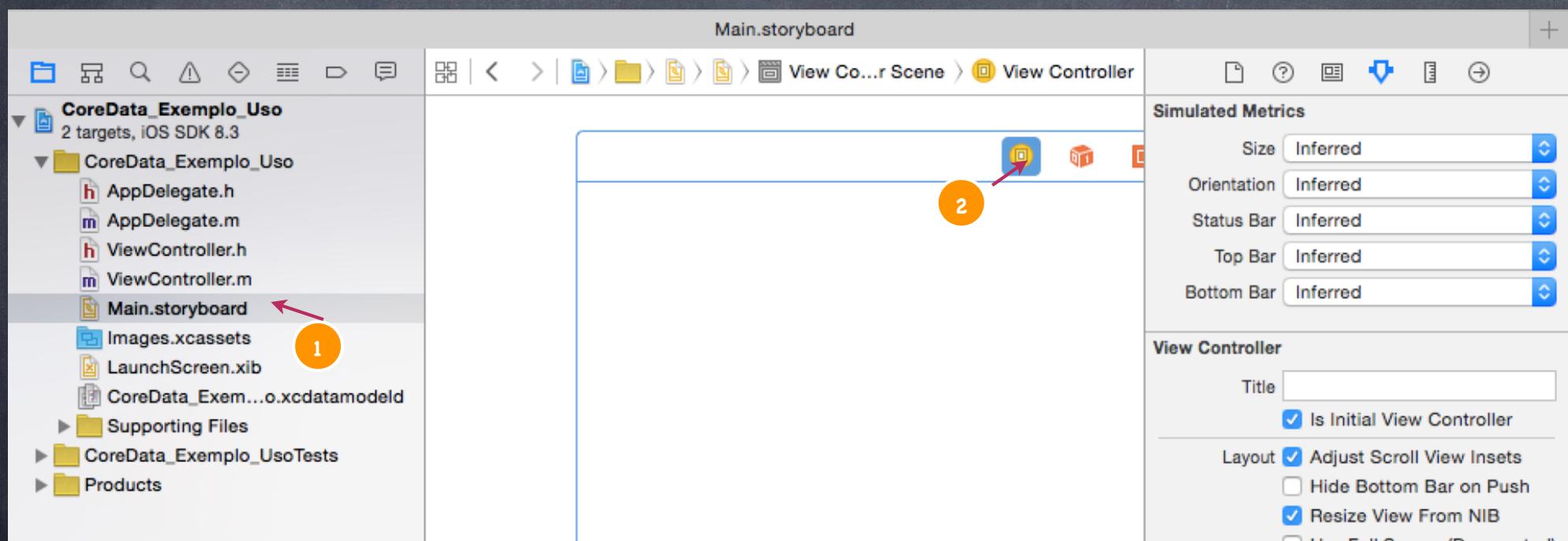
Core Data

- Nomeie o projeto como “CoreData_ExemploUso”(1), escolha o device iPhone(2), marque o checkbox (3).



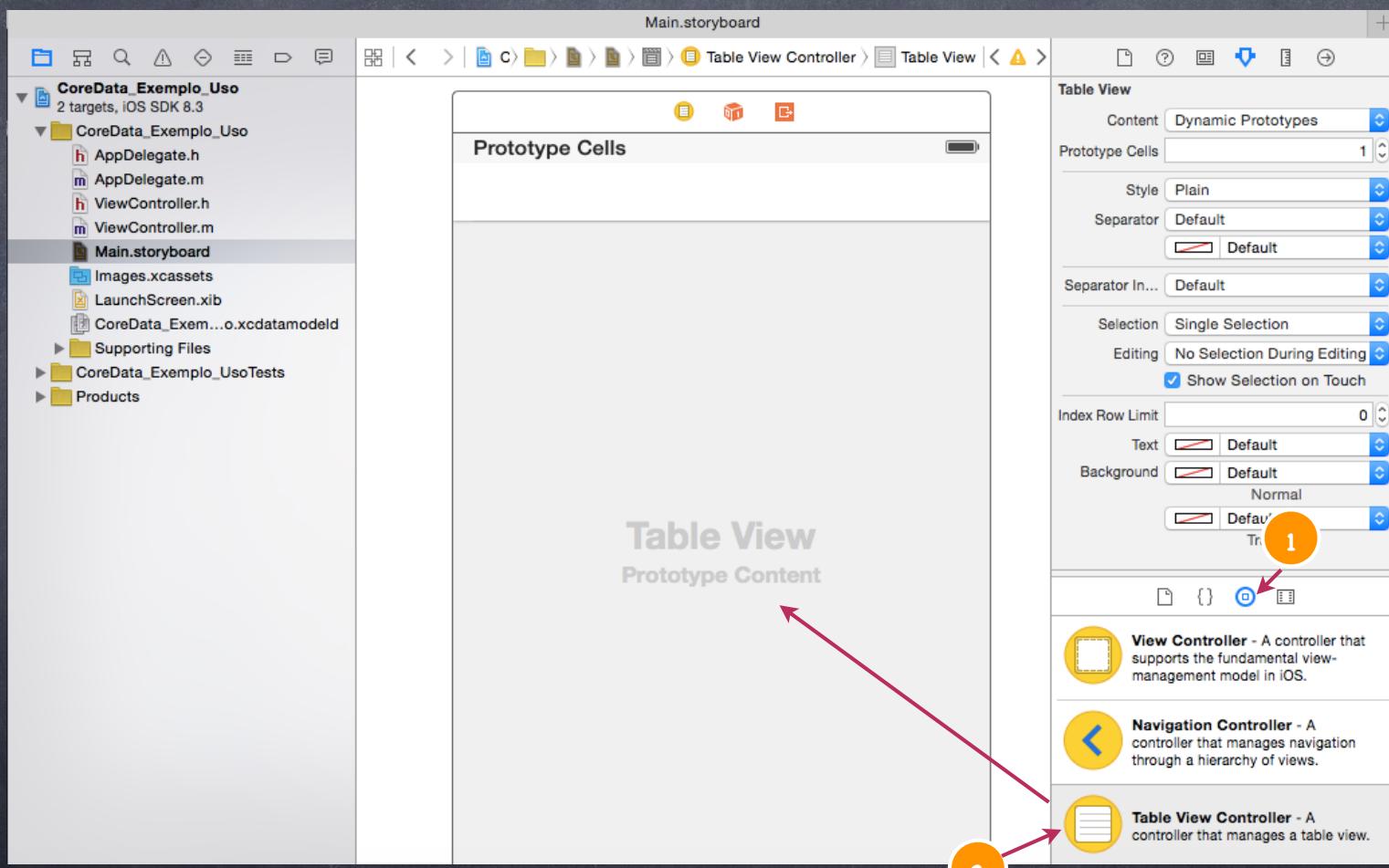
Core Data

- Vá em StoryBoard(1) selecione a viewController(2) e a apague, após apagar esta view, mesmo que ao incluirmos outras telas precisaremos ligar a View a um Controller pelo identity inspector, faremos essa ligação depois.



Core Data

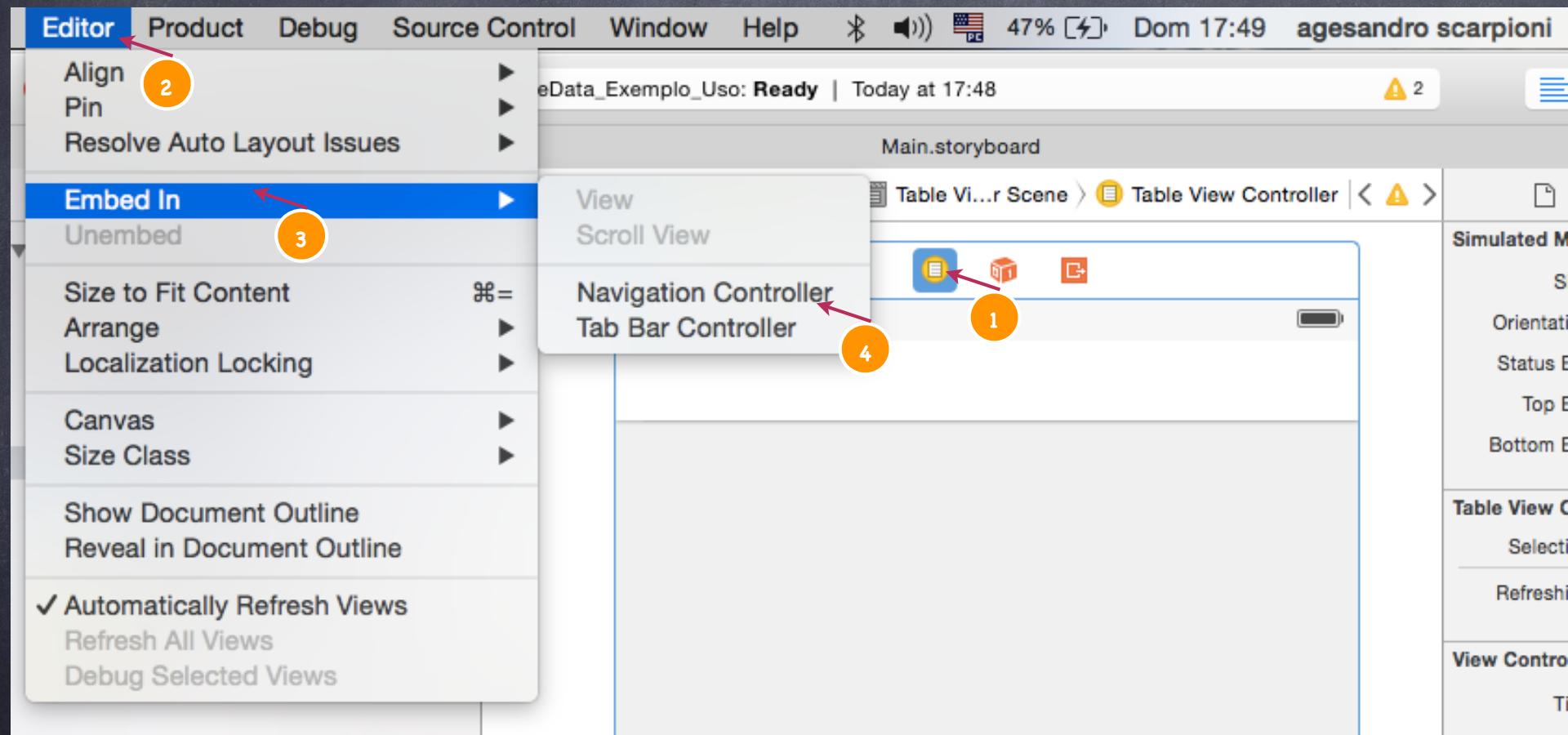
- Em Object Library (1), coloque um UITableViewController(2) no local de onde foi retirado o ViewController.



Dica: Vamos colocar no próximo Slide o Navigator Controller ele vai ficar antes da View usando o menu → Editor → Embed In - Navigation Controller.

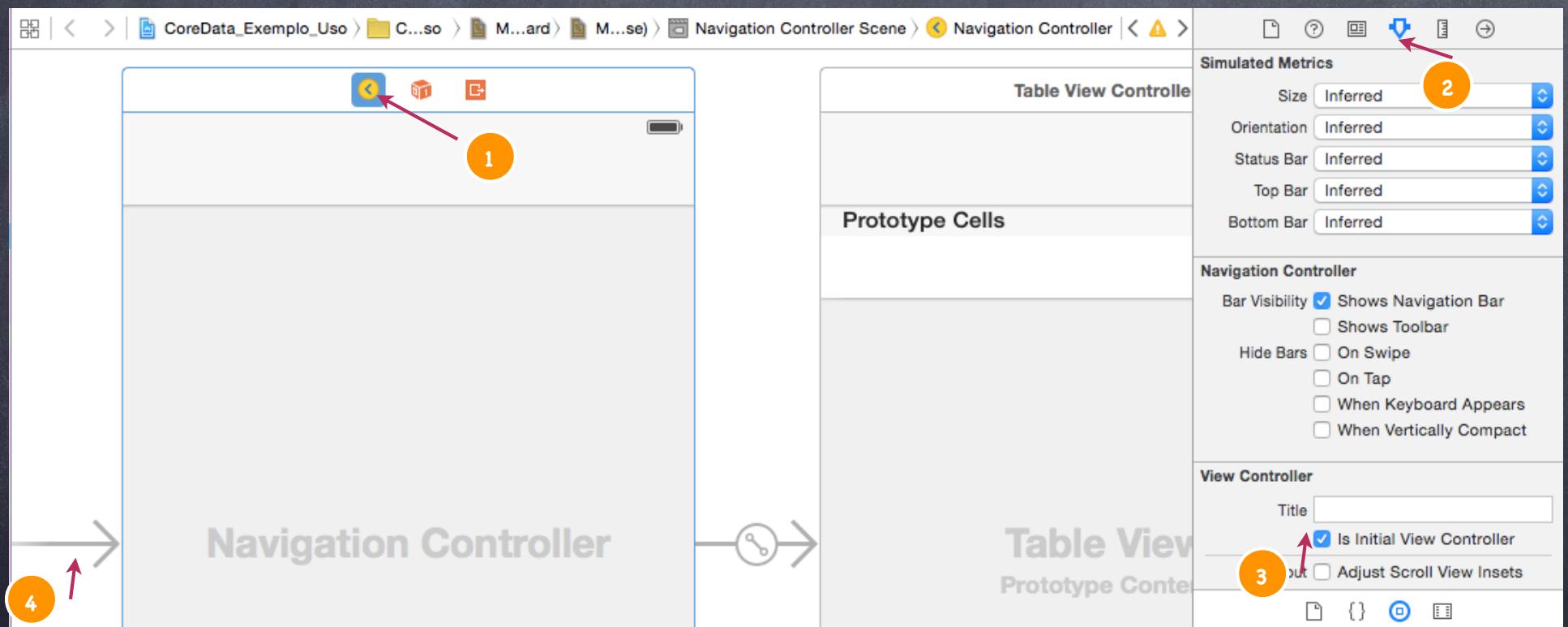
Core Data

- Com a UITableViewController selecionada (1), Clique em Editor(2) -> Embed In(3) -> Navigation Controller(4).



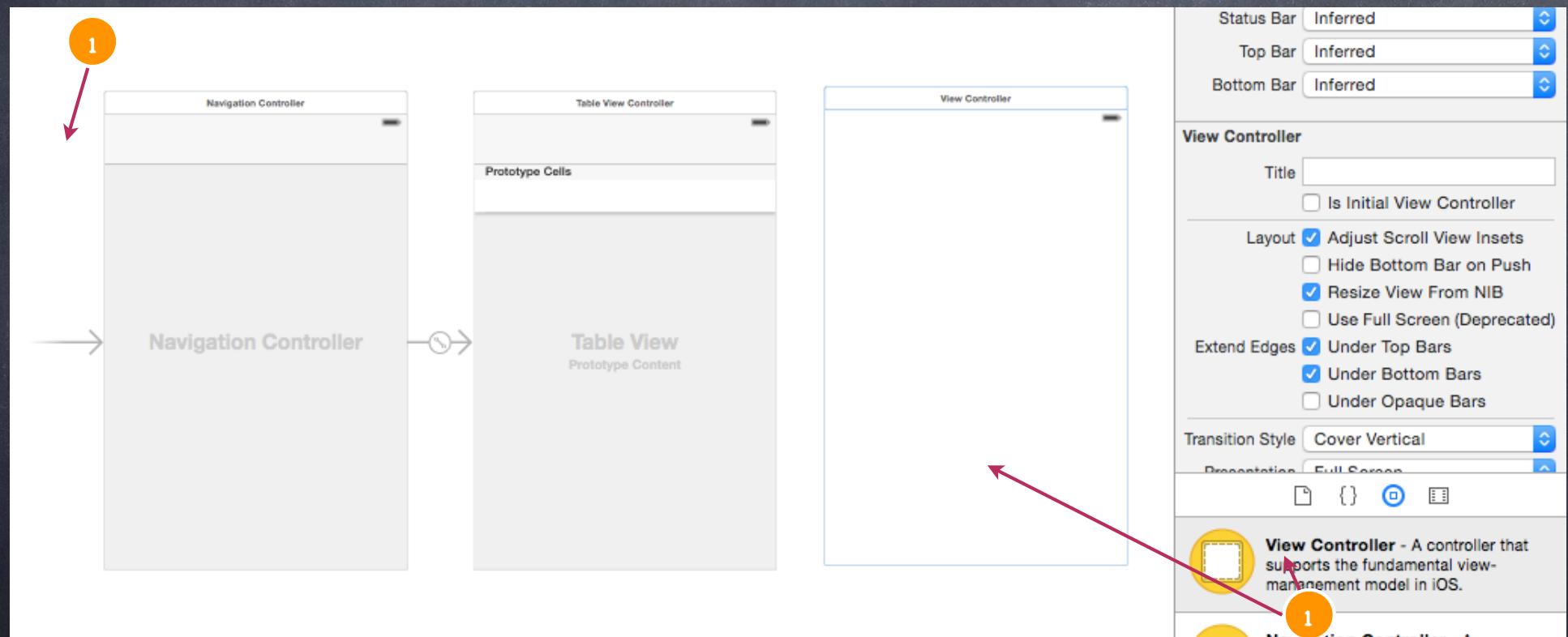
Core Data

- Com a Navigation Controller selecionado (1), Clique em Atributes Inspector e marque a opção do Checkbox (3) - Is Inicial View Controller. Isto fará aparecer a seta (4) para definir qual tela irá aparecer quando executarmos o programa.



Core Data

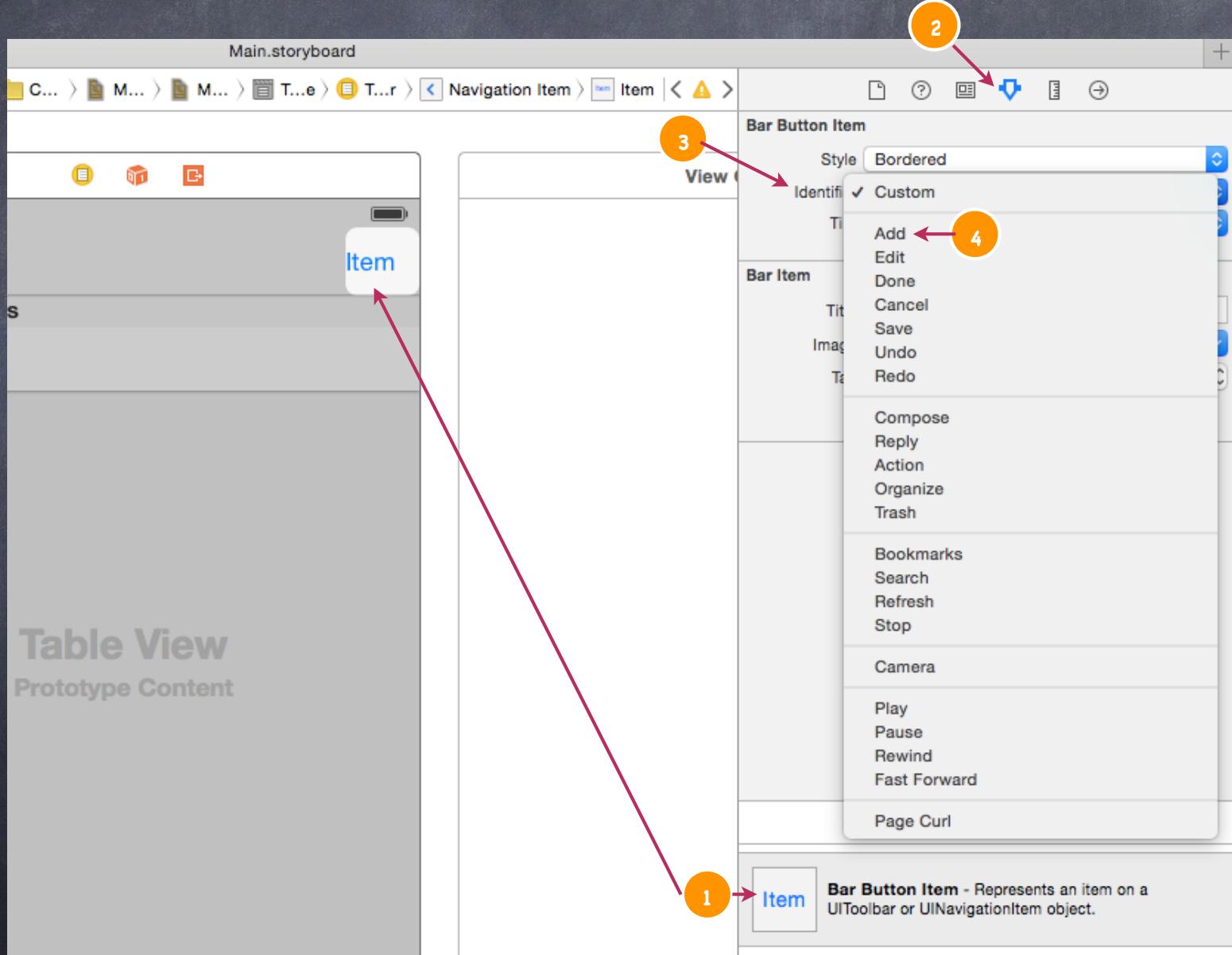
- Automaticamente o Navigation Controller é ligado ao TableView Controller, diminua o Zoom, clicando 2x em qualquer parte da área branca (1), em seguida inclua um ViewController (2).



Core Data

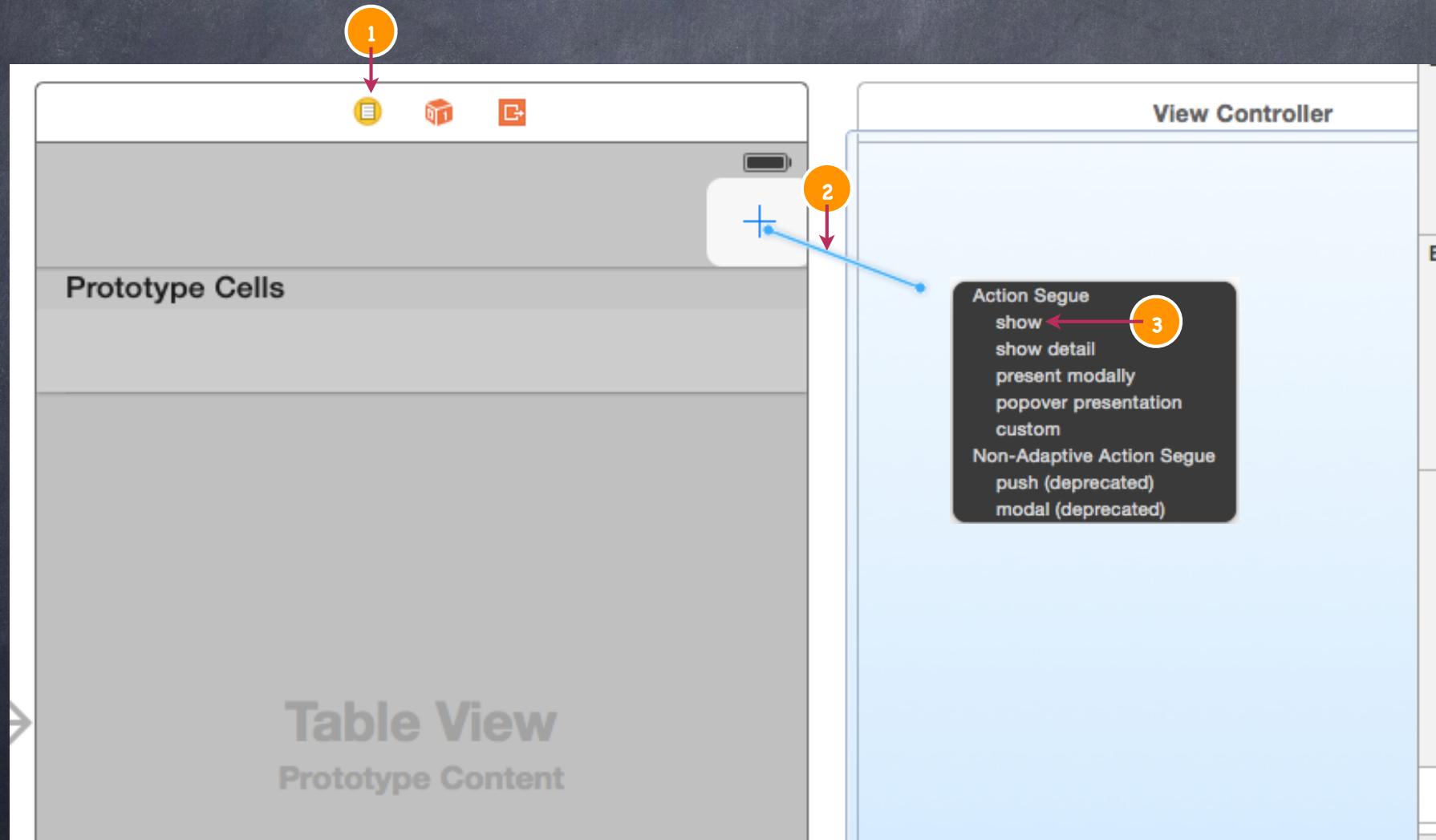
FIAP

- Aumente o Zoom e insira um Bar Button (1) no UITableViewController, vá em Attributes Inspector(2) e altere a identificação (3) para Add (4).



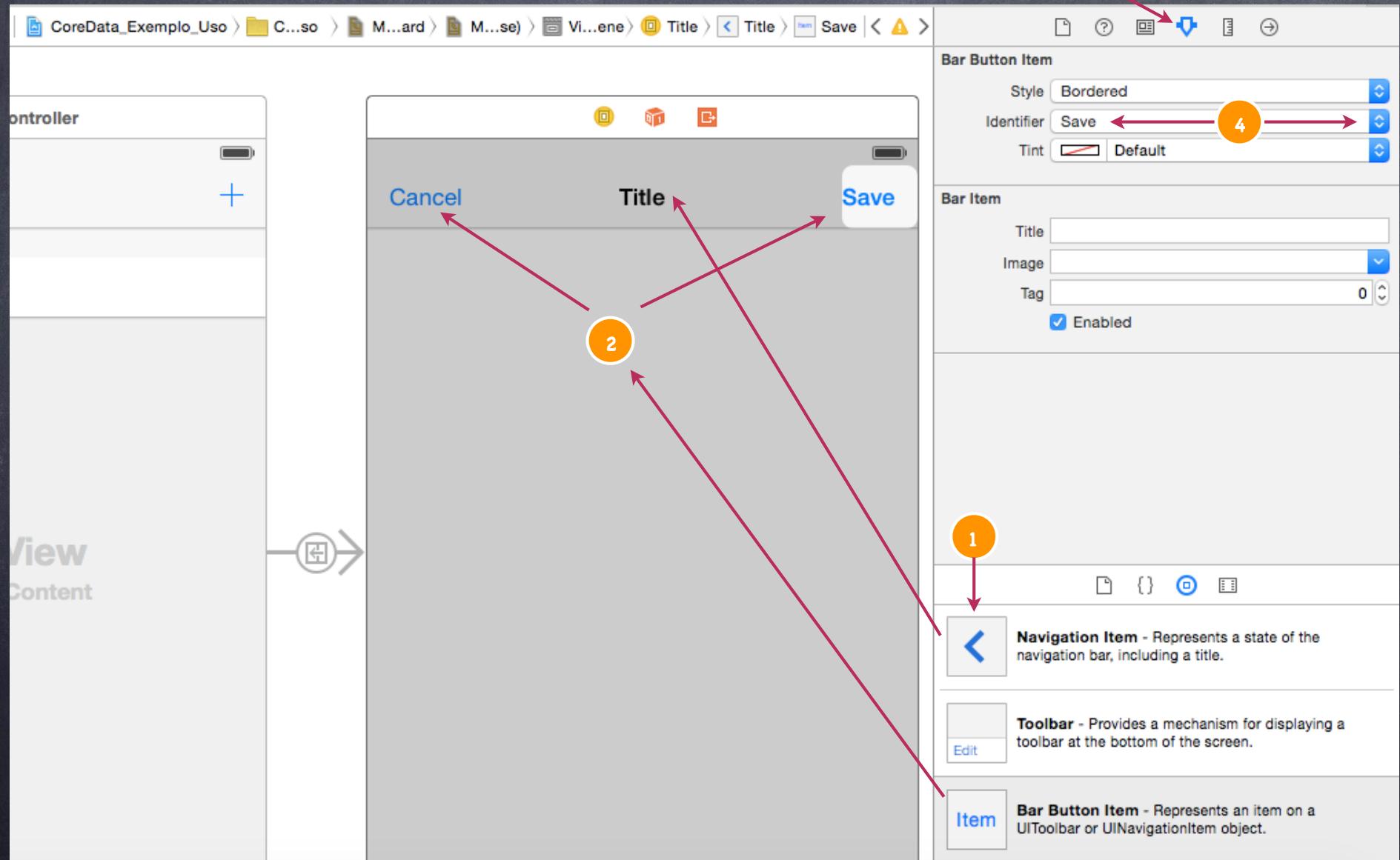
Core Data

- Com selecione o botão Add (1) e com o botão direito do mouse arraste até ao ViewController (2), escolha Show(3).



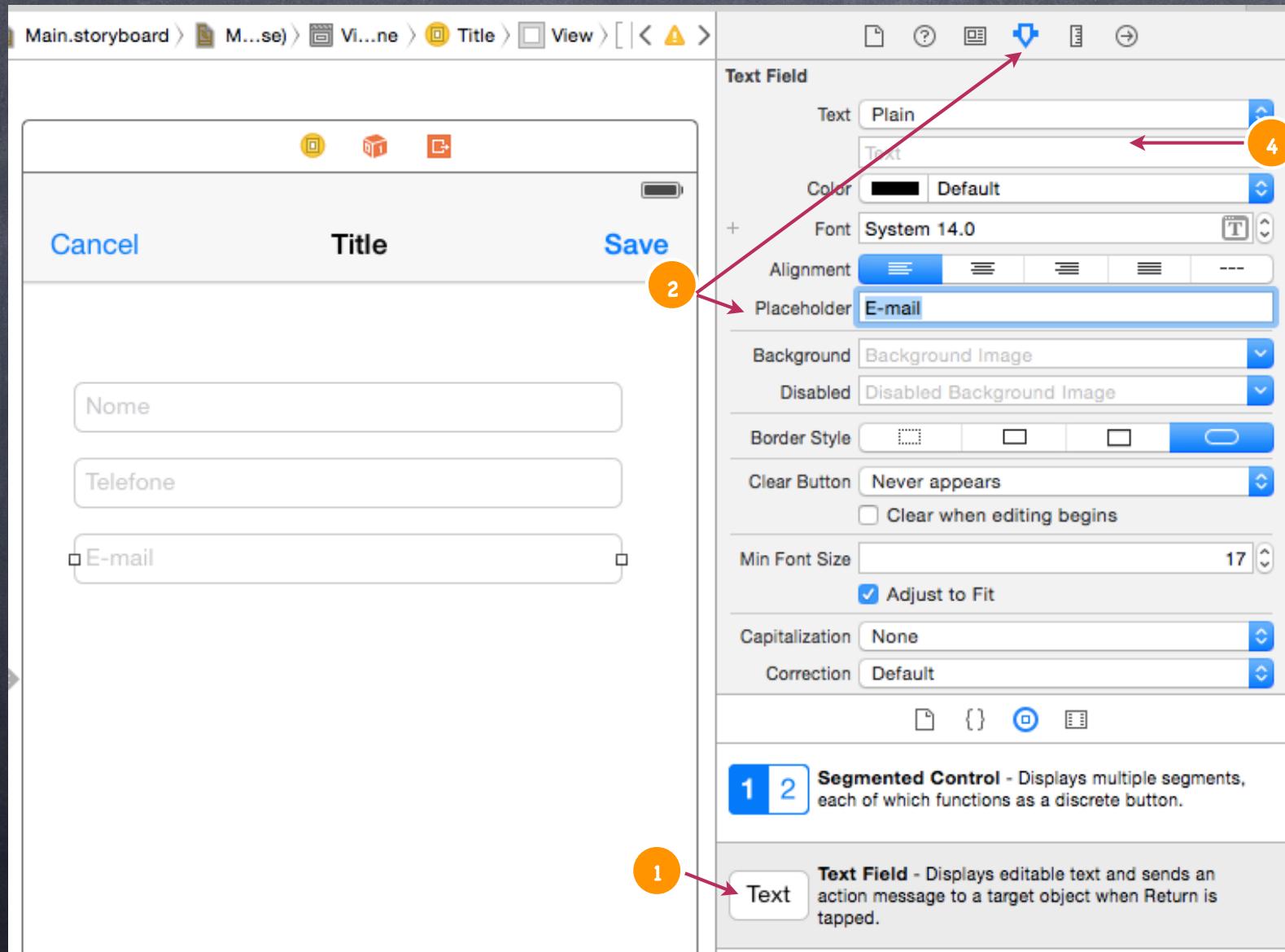
Core Data

- Insira no ViewController um Navigation Item(1), dois bar Buttons(2), no Identifier (4), altere o primeiro para Cancel e o segundo para Save.



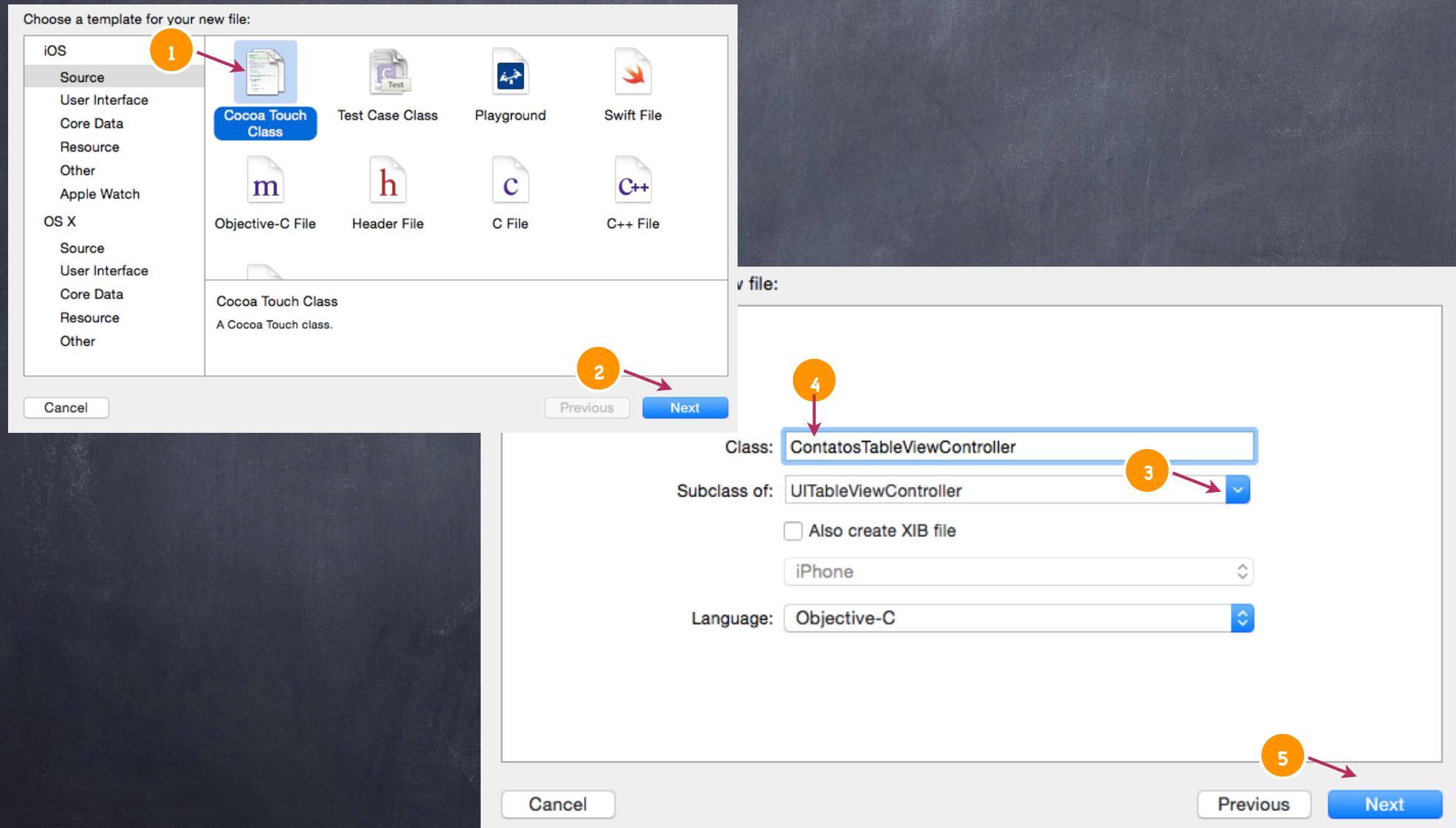
Core Data

• Insira no ViewController três Text's e altere a propriedade Placeholder de cada um.



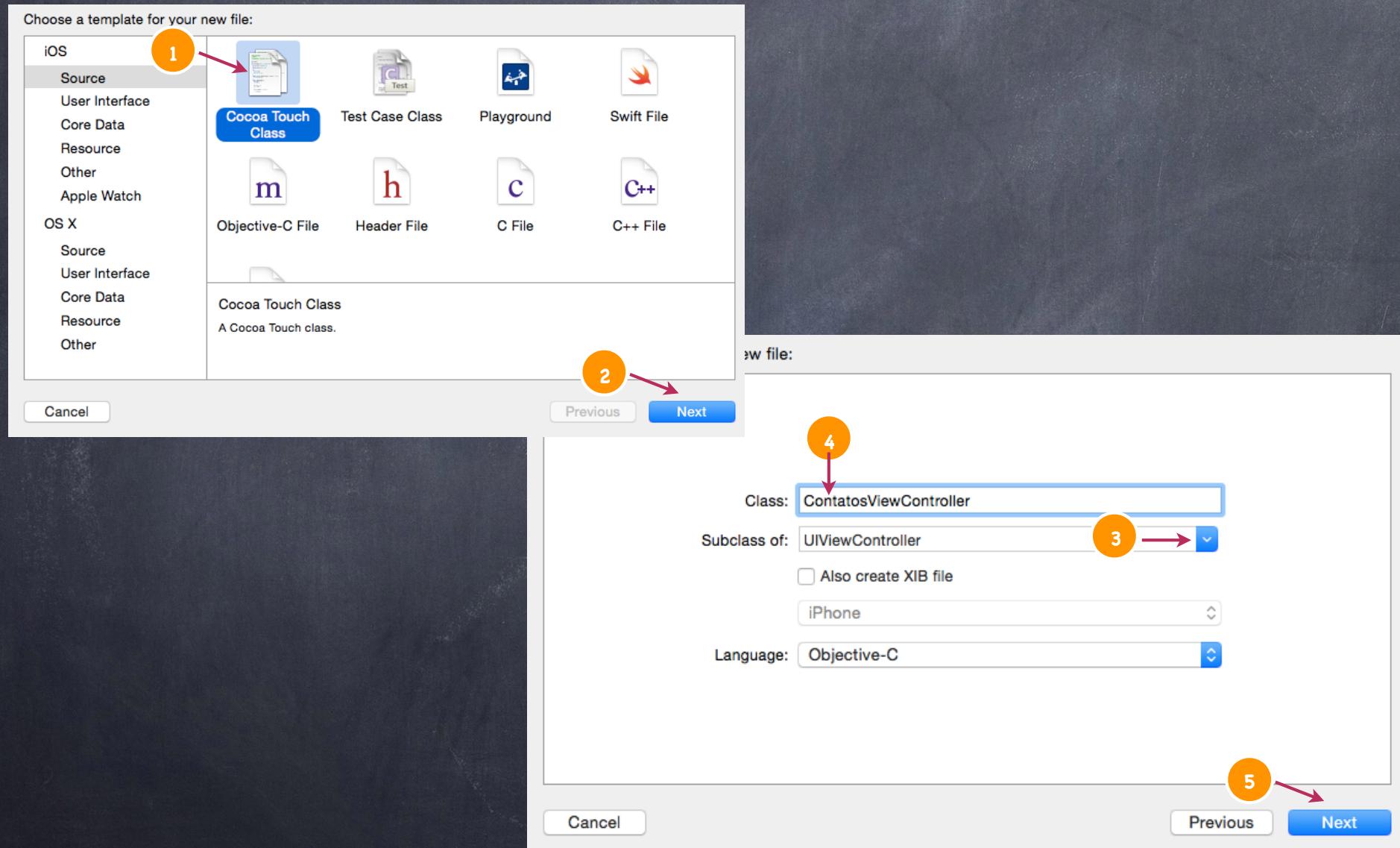
Core Data

- Adicione uma nova Classe (Command + N), escolha subclasse de UITableViewController e nomeie como ContatosTableViewController.



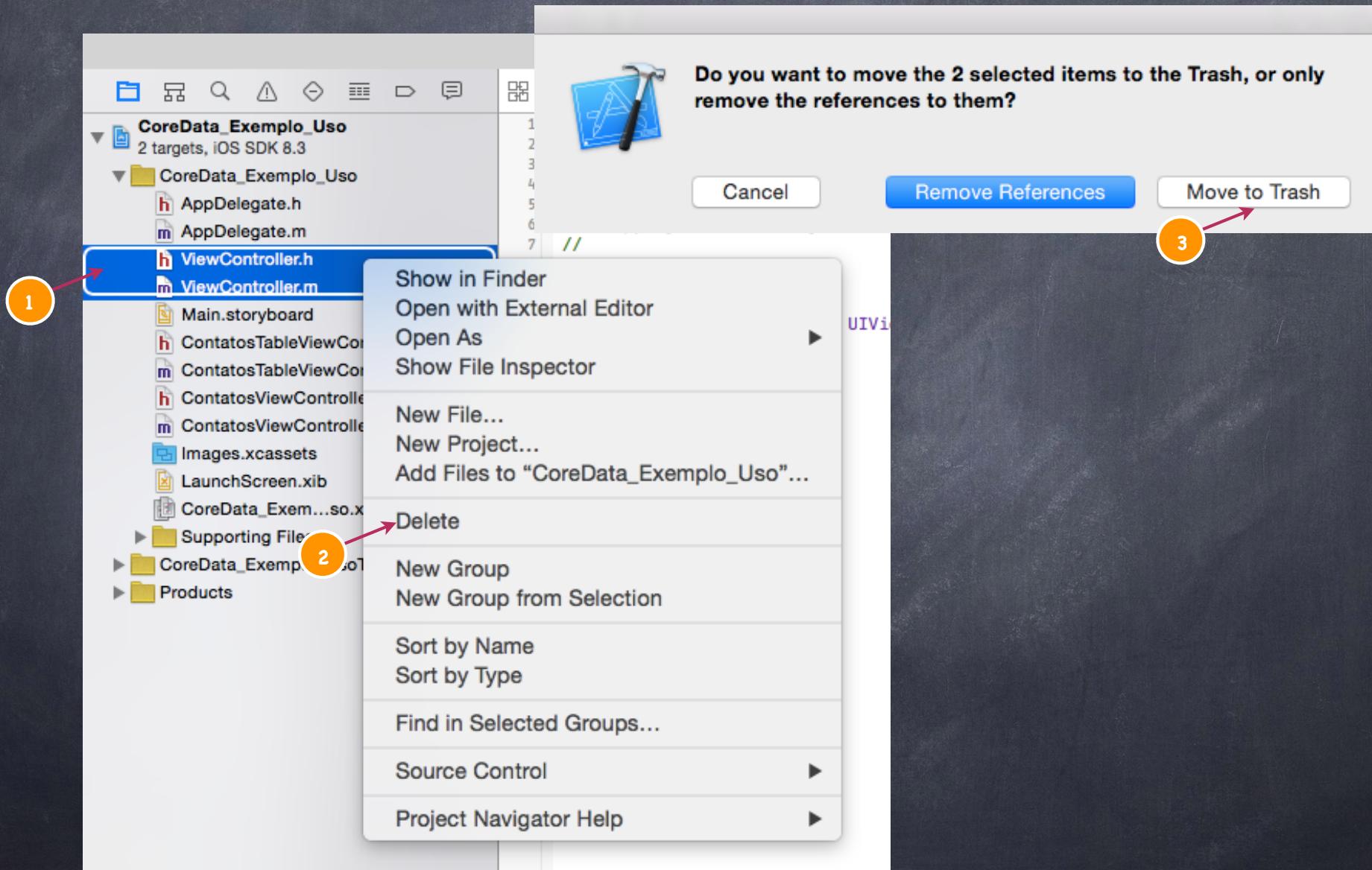
Core Data

- Adicione outra Classe, subclasse de UIViewController e nomeie como ContatosViewController.



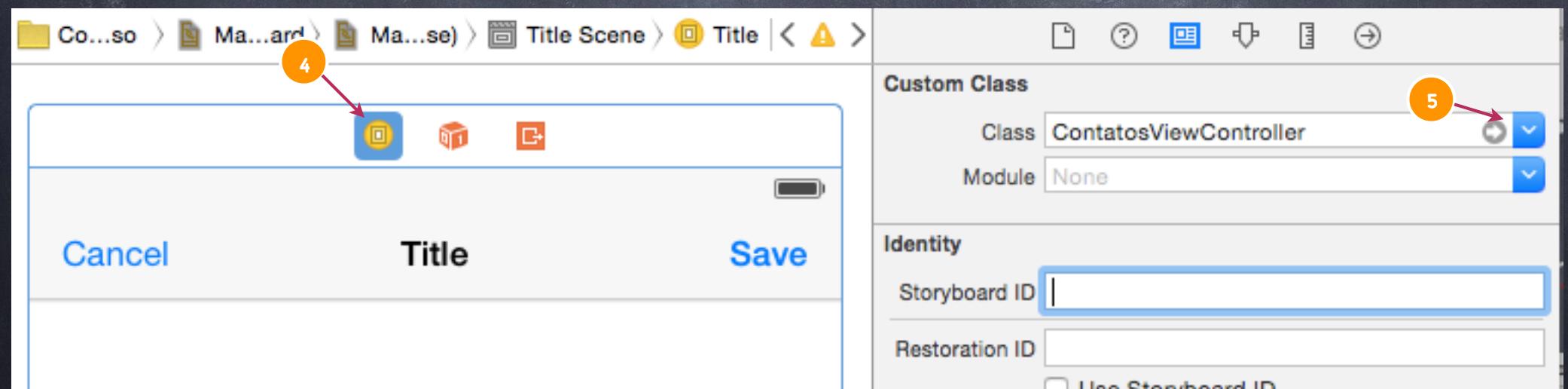
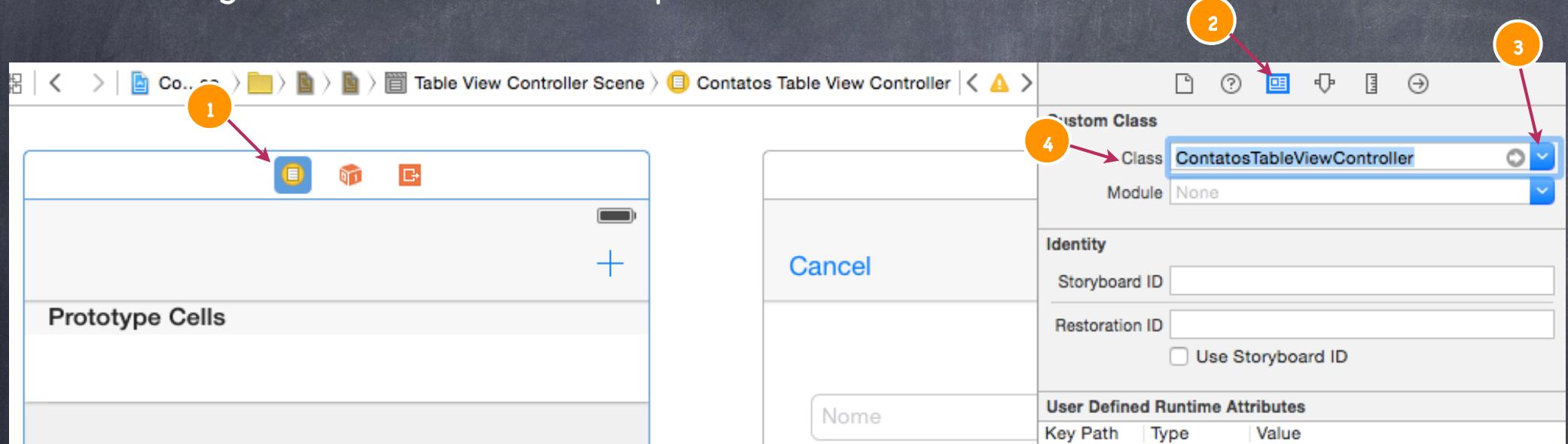
Core Data

- Selecione a ViewController.h e .m que foram criadas junto com o projeto, com o botão direito escolha delete e clique em Move To Trash.



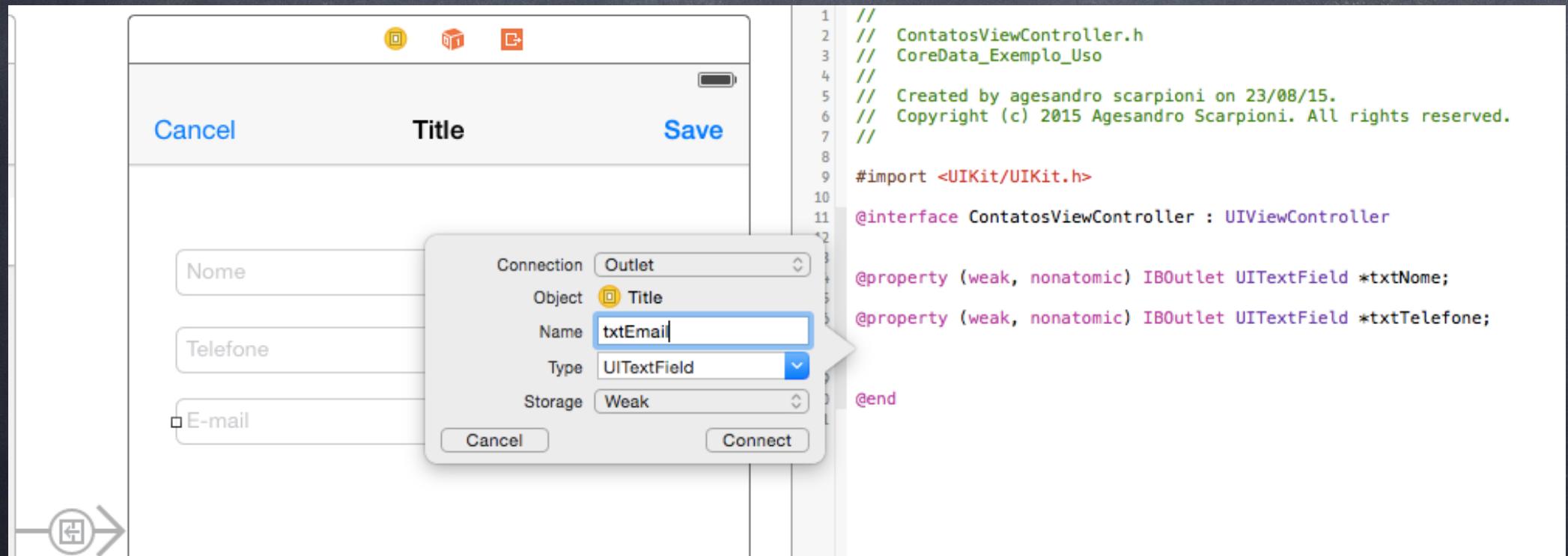
Core Data

• Vamos ligar as Views aos seus respectivos controllers.



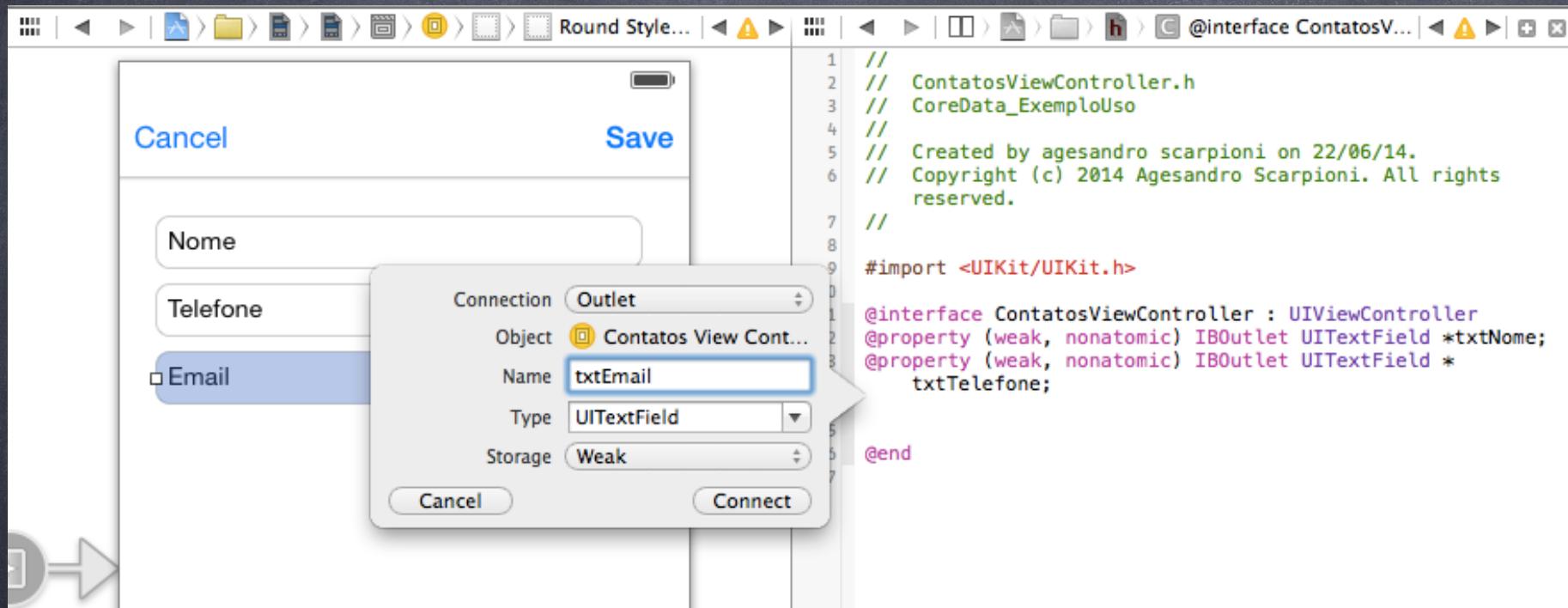
Core Data

- Vamos deixar a tela aberta simultaneamente no ContatosViewController.h e a Viewcontroller onde existem as caixas de texto, para criarmos as 3 propertys das caixas de texto.



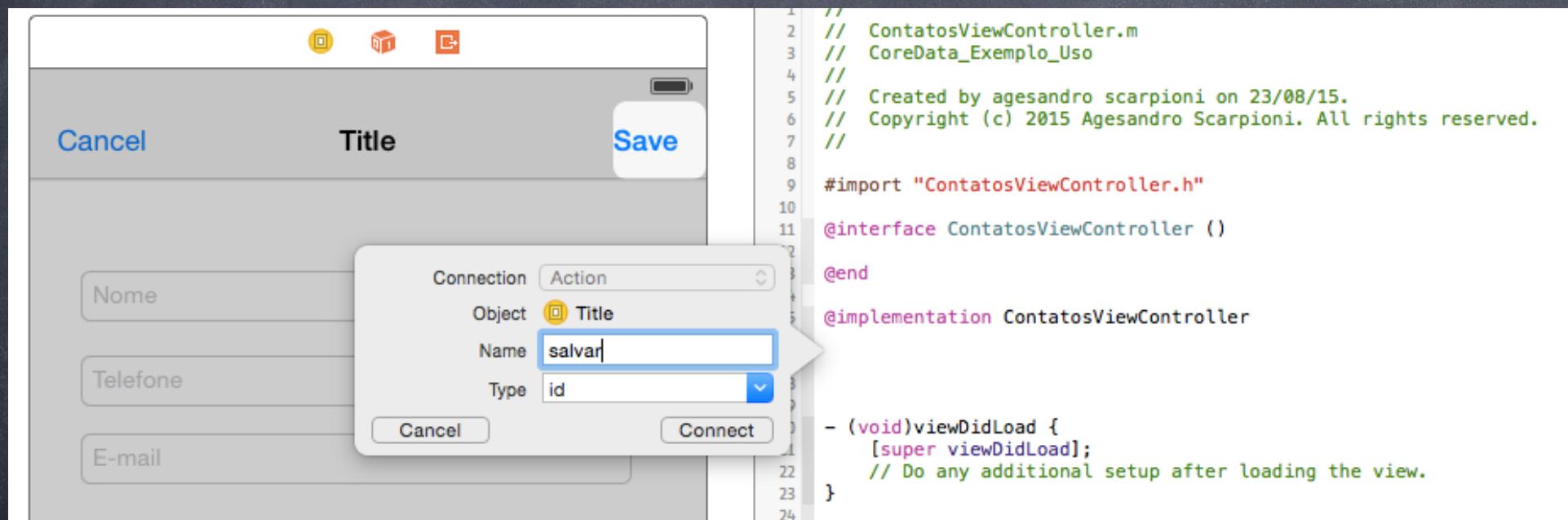
Core Data

- Use como sugestão de nomes para os IBOutlets txtNome, txtTelefone e TxtEmail.



Core Data

- No ContatosViewController.m crie os IBActions para os botões Cancel e Save, use os nomes cancelar e salvar, optei por colocar direto no arquivo.m pois não pretendo importar essa classe e por isso não preciso dos métodos públicos, eles podem ser locais.



Core Data

- Veja seus IBAction's como no exemplo abaixo(1), aproveite para fazer o synthesize(2) das três caixas de texto.

```
1 //  
2 // ContatosViewController.m  
3 // CoreData_ExemploUso  
4 //  
5 // Created by agesandro scarpioni on 22/06/14.  
6 // Copyright (c) 2014 Agesandro Scarpioni. All rights  
reserved.  
7 //  
8  
9 #import "ContatosViewController.h"  
10  
11 @interface ContatosViewController ()  
12  
13 @end  
14  
15 @implementation ContatosViewController  
16 @synthesize txtEmail,txtNome,txtTelefone;  
17  
18 - (IBAction)salvar:(id)sender {  
19 }  
20  
21 - (IBAction)cancelar:(id)sender {  
22 }  
23  
24  
25
```



Core Data

- Programe no cancelar a linha responsável para voltar para o navigationController.

```
20
21 - (IBAction)cancelar:(id)sender {
22     [self.navigationController popViewControllerAnimated:YES];
23 }
24
```

- Execute sua aplicação (command + R) e observe que o botão + e o botão cancelar já estão funcionando.

ATENÇÃO

ESTA PARTE É APENAS PARA QUEM NÃO CLICOU NO CHECKBOX
COREDATA NO INÍCIO DO PROJETO, CASO VOCÊ TENHA CLICADO VÁ PARA O
SLIDE 31

Core Data

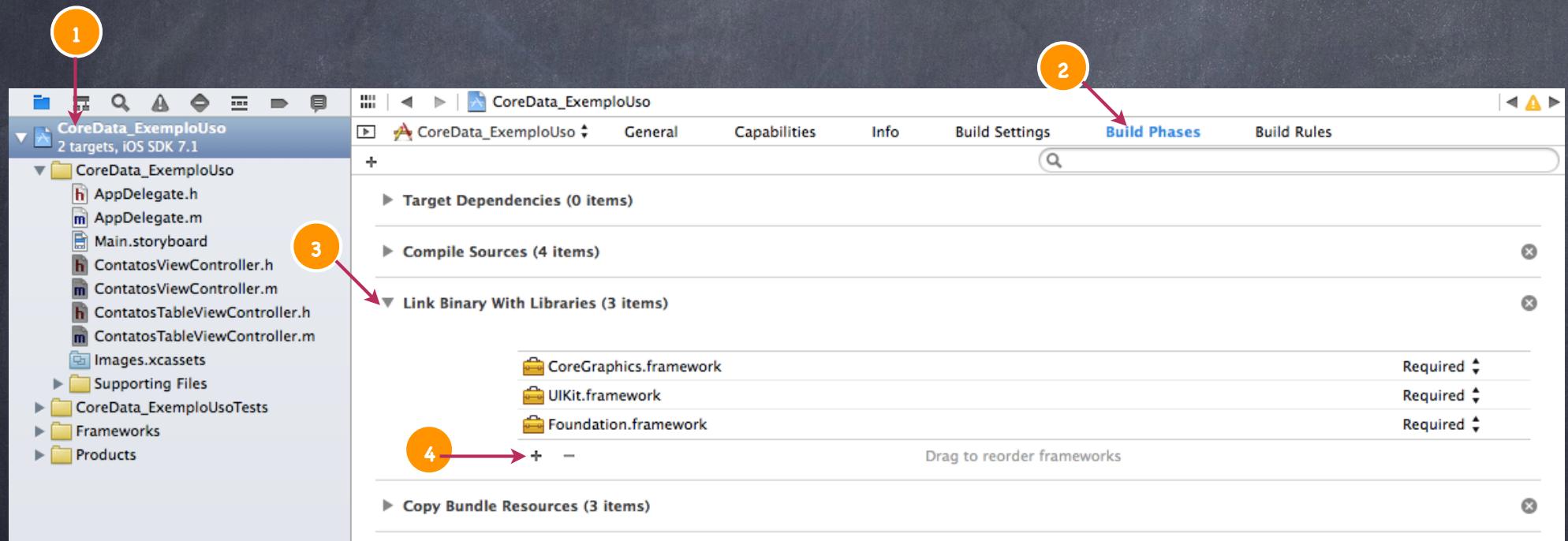
Inserindo a biblioteca Core Data e
preparando os atributos para armazenamento - Parte 2

X-Code

Prof. Agesandro Scarpioni

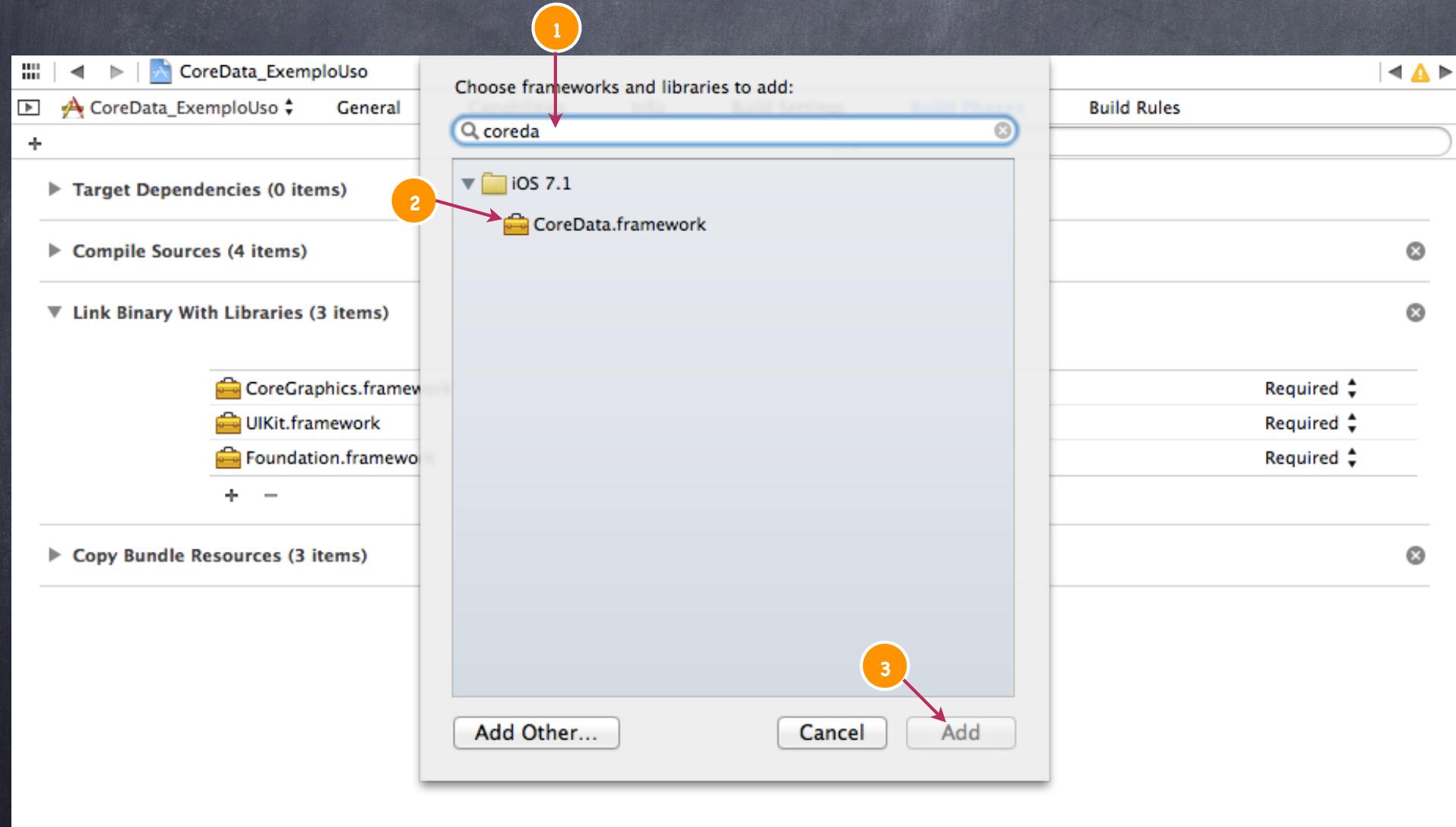
Core Data

- Primeiro passo para adicionar a biblioteca Core Data é clicar no projeto (1), clicar em Build Phases(2) abrir o link das bibliotecas (3) e clicar no botão (+) (4).



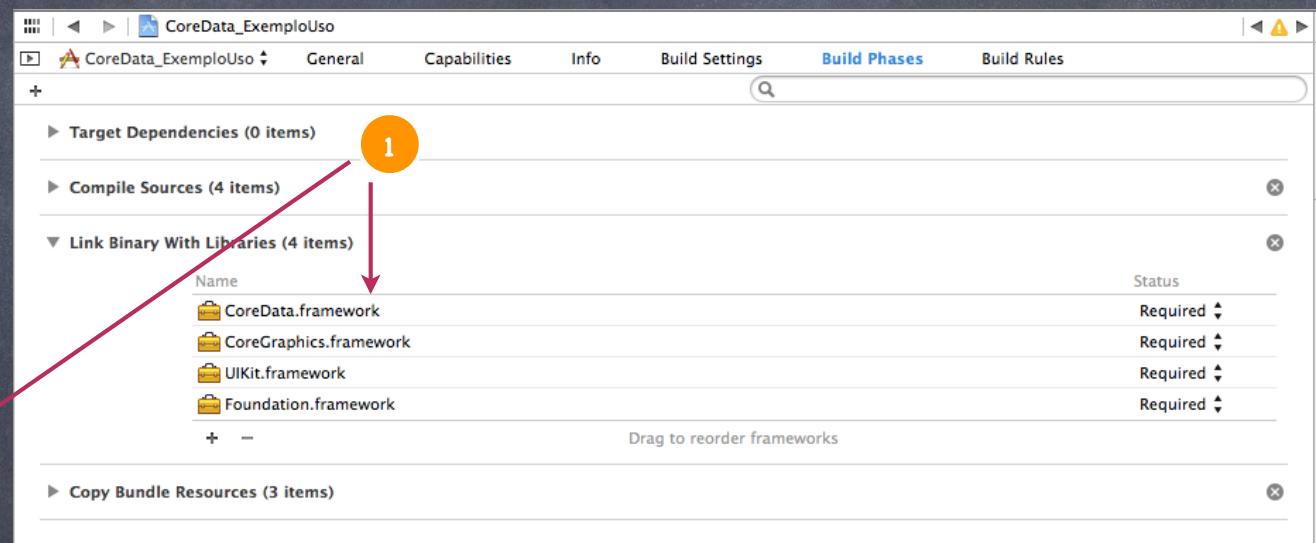
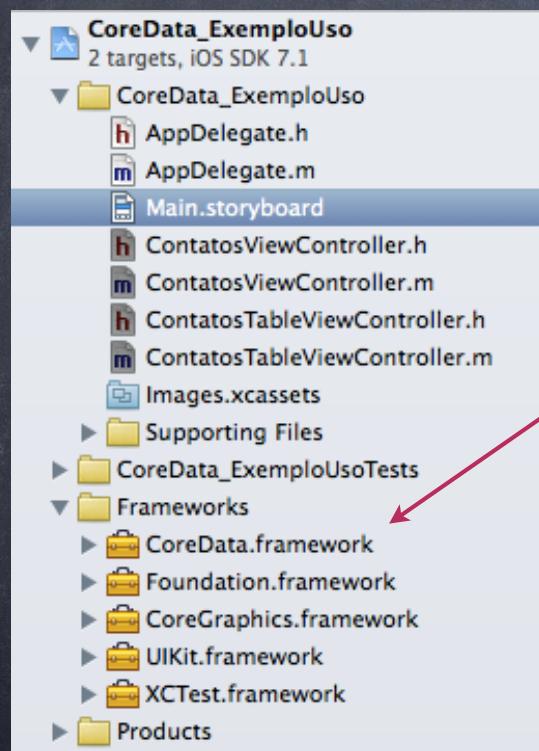
Core Data

• Digite CoreData... na tela que irá aparecer, dessa forma você filtra as bibliotecas.



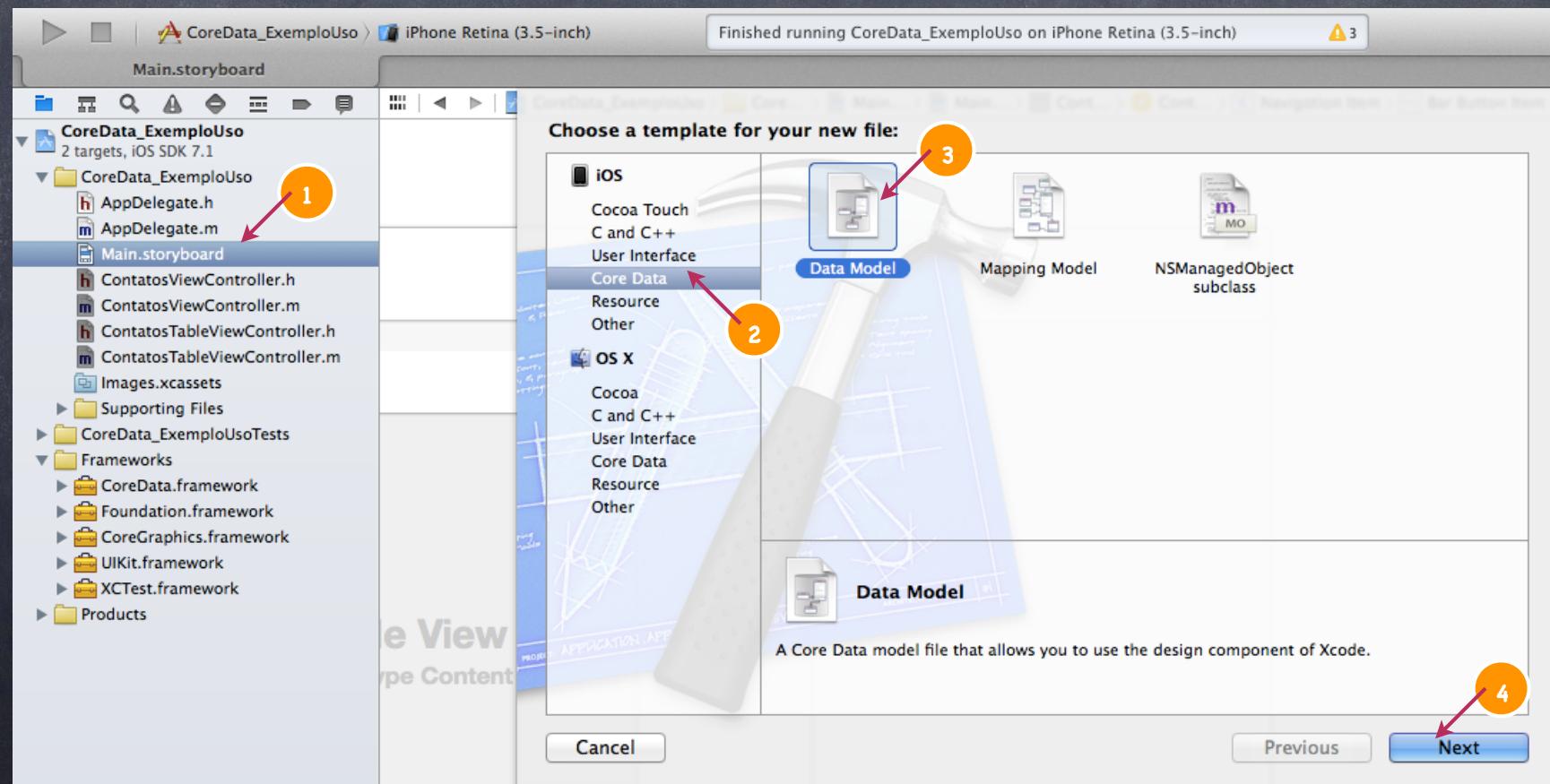
Core Data

Veja que a biblioteca foi adicionada.



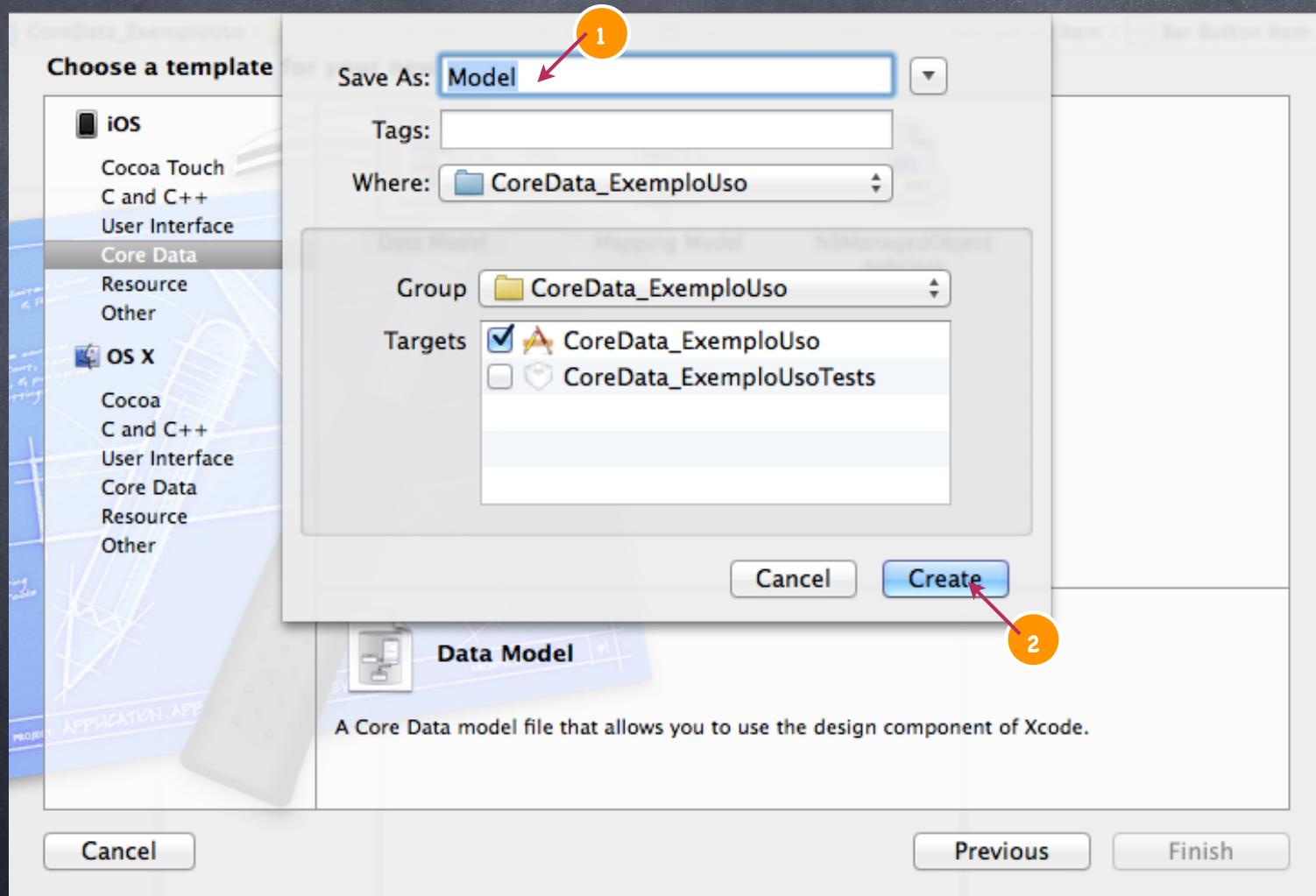
Core Data

- Agora que o framework foi adicionado ao projeto, precisamos criar o arquivo que faz a configuração e o mapeamento das entidades. Esse arquivo é chamado de Data Model e nele são mapeados uma ou mais entidades. Para cada entidade é necessário informar quais os atributos que precisam ser persistidos e os seus respectivos tipos(int, string, etc). O Core Data irá ler todos os dados necessários e gerar os SQL's em tempo de execução. Para criar seu Data Model clique no StoryBoard (1) e dê um Command + N.



Core Data

- Nomeie como Model e clique em Create, clicamos no StoryBoard apenas para organizar o código, o Model aparecerá bem abaixo do StoryBoard como será visto no próximo Slide.



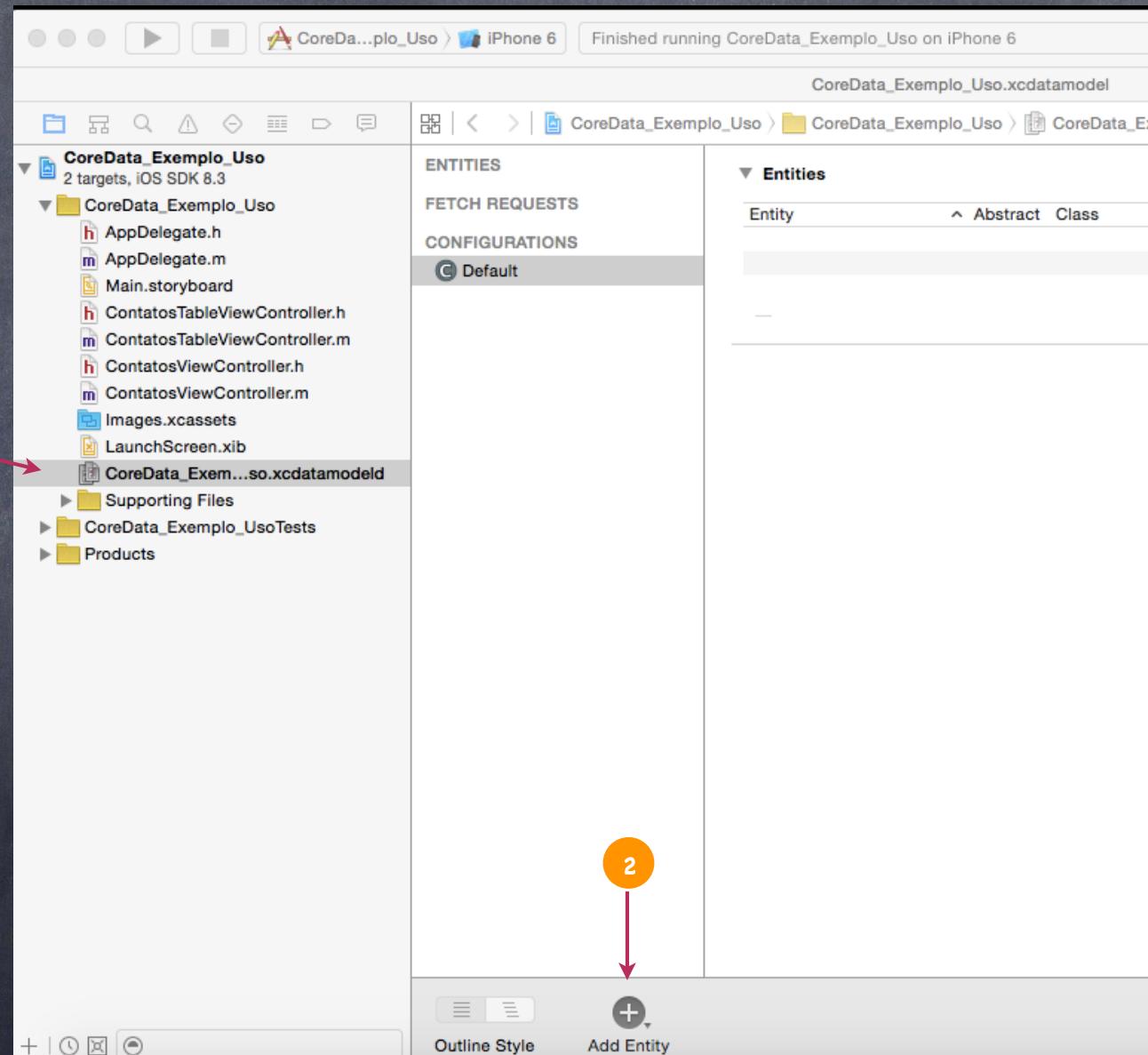
Core Data

- ⦿ A partir desse ponto vamos criar a entidade Contato. (Esta entidade irá aparecer em nosso código como um NSManagedObject ou suas sub classes)
- ⦿ Criar os atributos: Email, Nome e Telefone todos como String (Atributos são acessados em nosso NSManagedObjects via os métodos valueForKey: e setValueForKey:.)
- ⦿ Se nossa sub classe é NSManagedObject, podemos acessar os atributos por @propertys.

Core Data

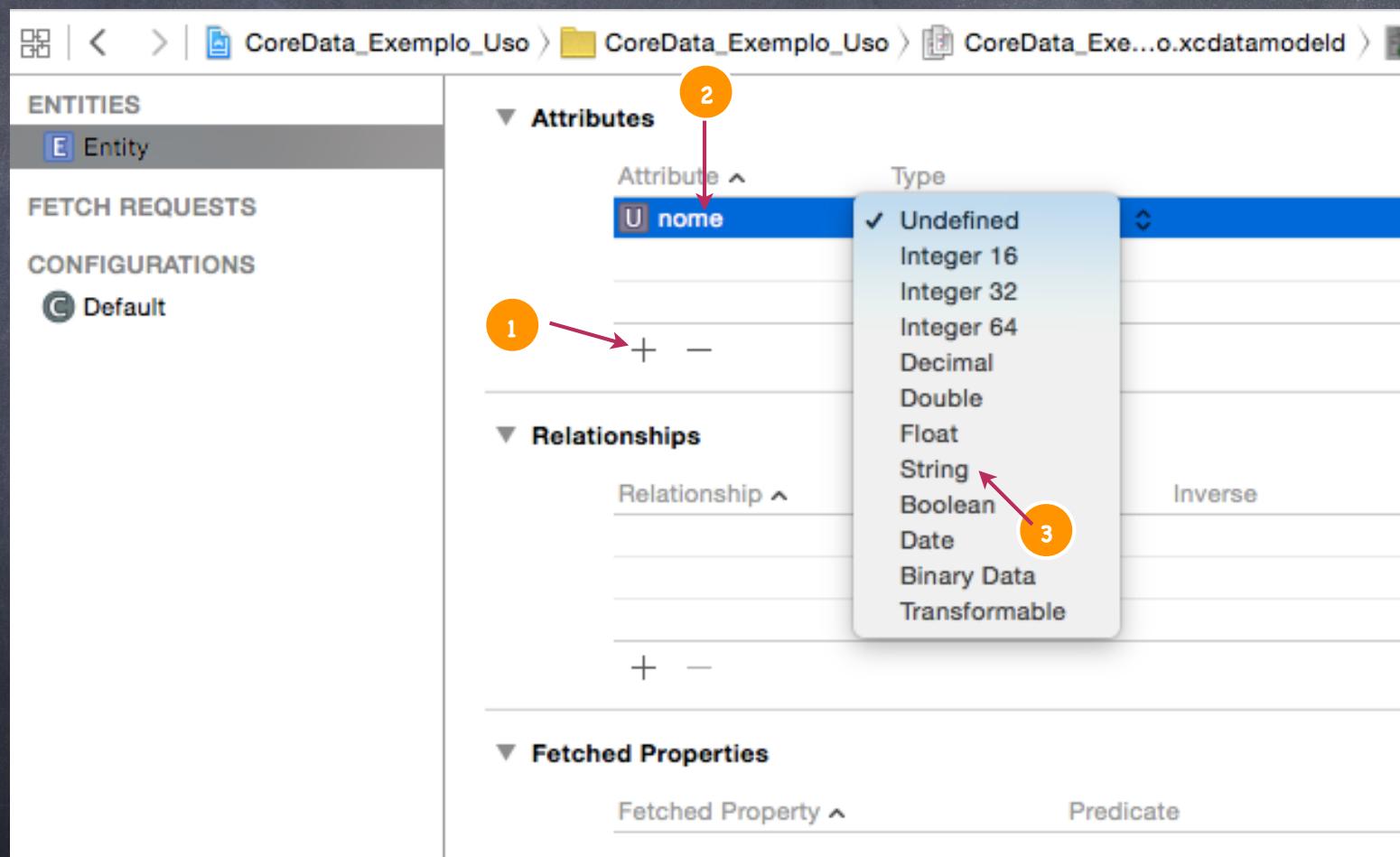
FIAP

- Veja o Model logo abaixo do Storyboard (1), para adicionar uma entidade clique no Add Entity (2).



Core Data

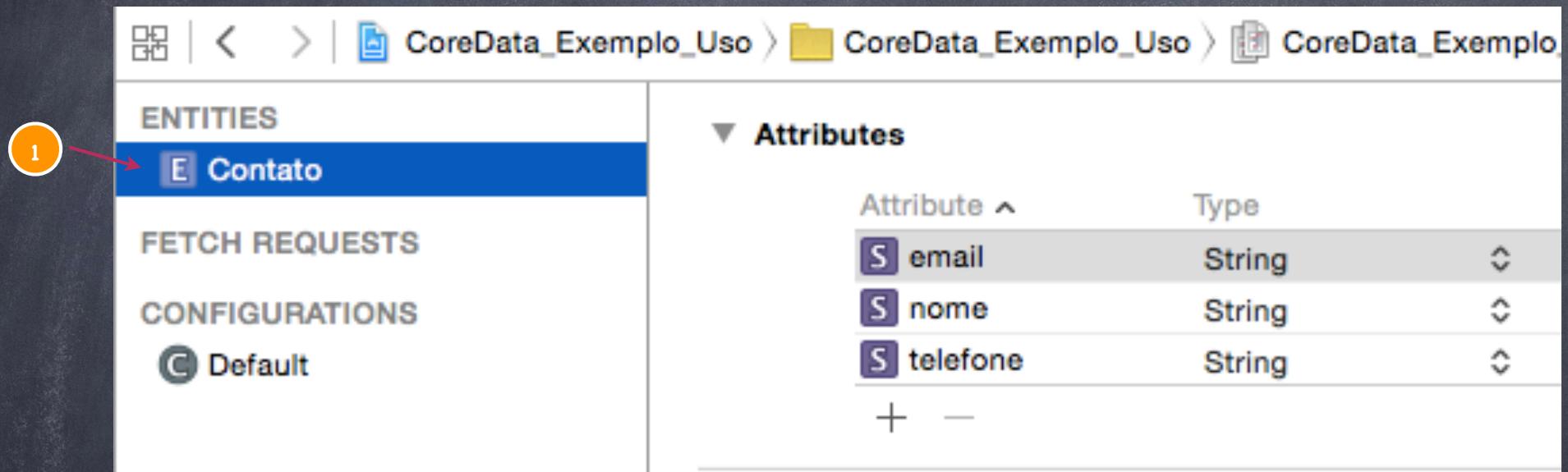
- Para adicionar os atributos clique em (+) (1), digite o nome do atributo exemplo: nome (2) e escolha String(3). Repita os passos para telefone e email.



Core Data

FIAP

- Após inserir todos os nomes dos atributos, troque o nome da Entity para Contato(1).



Core Data

Preparando o projeto para o Core Data - Parte 3

X-Code

Prof. Agesandro Scarpioni

Core Data

FIAP

- AppDelegate.h, faça o import do Core Data e digite as 3 linhas de @property e o NSURL, caso já tenha selecionado o Core Data no início do projeto não precisa criar as linhas abaixo. Leia a OBS no final do código.

```
1 //  
2 // AppDelegate.h  
3 // CoreData_Exemplo_Uso  
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 #import <CoreData/CoreData.h>  
11  
12 @interface AppDelegate : UIResponder <UIApplicationDelegate>  
13  
14 @property (strong, nonatomic) UIWindow *window;  
15 //A linha abaixo é o contexto de persistência de Objetos  
16 @property (readonly, strong, nonatomic) NSManagedObjectContext *managedObjectContext;  
17 //A linha abaixo é a instância do arquivo Managed Object Model que é o arquivo ???xcdatamodeld  
18 // onde ??? é o nome do seu projeto ou o nome informado quando criamos o Model.  
19 @property (readonly, strong, nonatomic) NSManagedObjectModel *managedObjectModel;  
20 //define o meio de campo entre os objetos e o mecanismo de persistência físico que é o SQLite.  
21 @property (readonly, strong, nonatomic) NSPersistentStoreCoordinator *persistentStoreCoordinator;  
22  
23 - (void)saveContext;  
24 //Retorna uma URL para o local onde o banco de dados será salvo.  
25 - (NSURL *)applicationDocumentsDirectory;  
26  
27  
28 /*  
29 OBS: IMPORTANTE IMPORTANTE IMPORTANTE  
30 O CÓDIGO QUE teremos que digitar no AppDelegate.h é muito extenso, ele faz parte do wizard de criação de  
31 projetos Xcode, quando clicamos em NEW --> PROJECT --> IOS --> Application --> e escolhemos um template  
32 é fornecido um checkbox para inserir o Core Data no projeto, gerando o código automaticamente.  
33 */  
34  
35 @end  
36  
37
```

Core Data

- No AppDelegate.m digite as 3 linhas de @synthesize. Se você clicou no CoreData essas linhas já estão prontas.

```
1 //////////////////////////////////////////////////////////////////
2 // AppDelegate.m
3 // CoreData_ExemploUso
4 //
5 // Created by agesandro scarpioni on 22/06/14.
6 // Copyright (c) 2014 Agesandro Scarpioni. All rights reserved.
7 //
8
9 #import "AppDelegate.h"
10
11 @implementation AppDelegate
12
13 //sintaxe _propriedade permite acessar como var local, sintaxe self.propriedade vai
14 //acessar o getter/setter.
15
16 @synthesize managedObjectContext = _managedObjectContext;
17 @synthesize managedObjectModel = _managedObjectModel;
18 @synthesize persistentStoreCoordinator = _persistentStoreCoordinator;
19
20
```

Core Data

- No AppDelegate.m, continue a programação. Se você clicou no CoreData essas linhas já estão prontas.

```
19 #pragma mark Core Data stack
20
21 /**
22  Returns the managed object context for the application.
23  If the context doesn't already exist, it is created and bound to the persistent store coordinator for the application.
24 */
25
26
27 - (NSManagedObjectContext *)managedObjectContext {
28
29     if (_managedObjectContext != nil) {
30         return _managedObjectContext;
31     }
32
33     NSPersistentStoreCoordinator *coordinator = [self persistentStoreCoordinator];
34     if (coordinator != nil) {
35         _managedObjectContext = [[NSManagedObjectContext alloc] init];
36         [_managedObjectContext setPersistentStoreCoordinator:coordinator];
37     }
38     return _managedObjectContext;
39 }
```

Core Data

- No AppDelegate.m, continue a programação. Se você clicou no CoreData essas linhas já estão prontas.

```
40
41  /**
42   Returns the managed object model for the application.
43   If the model doesn't already exist, it is created from the application's model.
44   O Managed Object Model é o Model.xcdatamodeld
45  */
46 - (NSManagedObjectModel *)managedObjectModel {
47
48     if (_managedObjectModel != nil) {
49         return _managedObjectModel;
50     }
51     //Atenção esse nome Model na linha abaixo é o nome do arquivo Model.xcdatamodeld
52     NSURL *modelURL = [[NSBundle mainBundle] URLForResource:@"Model" withExtension:@"momd"];
53     _managedObjectModel = [[NSManagedObjectModel alloc] initWithContentsOfURL:modelURL];
54     return _managedObjectModel;
55 }
```

Core Data

- No AppDelegate.m, continue a programação, caso prefira existe uma imagem maior no próximo Slide.

```
56 /**
57  * Returns the persistent store coordinator for the application.
58  * If the coordinator doesn't already exist, it is created and the application's store added to it.
59  * Aqui é definido o nome do arquivo que vai conter o banco de dados
60  */
61 - (NSPersistentStoreCoordinator *)persistentStoreCoordinator {
62
63     if (_persistentStoreCoordinator != nil) {
64         return _persistentStoreCoordinator;
65     }
66     //Aqui estamos definindo o nome do arquivo de banco de dados -> ContatosCoreData.sqlite
67     NSURL *storeURL = [[self applicationDocumentsDirectory] URLByAppendingPathComponent:@"ContatosCoreData.sqlite"];
68
69     NSError *error = nil;
70     _persistentStoreCoordinator = [[NSPersistentStoreCoordinator alloc] initWithManagedObjectModel:[self managedObjectModel]];
71     if (![_persistentStoreCoordinator addPersistentStoreWithType:NSSQLiteStoreType configuration:nil URL:storeURL options:nil
72             error:&error]) {
73
74     /*
75      Replace this implementation with code to handle the error appropriately.
76
77      abort() causes the application to generate a crash log and terminate. You should not use this function in a shipping
78      application, although it may be useful during development. If it is not possible to recover from the error, display an
79      alert panel that instructs the user to quit the application by pressing the Home button.
80
81      Typical reasons for an error here include:
82      * The persistent store is not accessible;
83      * The schema for the persistent store is incompatible with current managed object model.
84      Check the error message to determine what the actual problem was.
85
86      If the persistent store is not accessible, there is typically something wrong with the file path. Often, a file URL is
87      pointing into the application's resources directory instead of a writeable directory.
88
89      If you encounter schema incompatibility errors during development, you can reduce their frequency by:
90      * Simply deleting the existing store:
91          [[NSFileManager defaultManager] removeItemAtURL:storeURL error:nil]
92
93      * Performing automatic lightweight migration by passing the following dictionary as the options parameter:
94          [NSDictionary dictionaryWithObjectsAndKeys:[NSNumber numberWithBool:YES], NSMigratePersistentStoresAutomaticallyOption,
95          [NSNumber numberWithBool:YES], NSInferMappingModelAutomaticallyOption, nil];
96
97      Lightweight migration will only work for a limited set of schema changes; consult "Core Data Model Versioning and Data
98      Migration Programming Guide" for details.
99
100     */
101    NSLog(@"Unresolved error %@", error, [error userInfo]);
102    abort();
103 }
104
105     return _persistentStoreCoordinator;
106 }
```

Core Data

- No AppDelegate.m, ATENÇÃO: A Mesma Programação sem os comentários com um tamanho maior de fonte. Se você clicou no CoreData essas linhas já estão prontas.

```
56
57 /**
58 Returns the persistent store coordinator for the application.
59 If the coordinator doesn't already exist, it is created and the application's store added to it.
60 Aqui é definido o nome do arquivo que vai conter o banco de dados
61 */
62 - (NSPersistentStoreCoordinator *)persistentStoreCoordinator {
63
64     if (_persistentStoreCoordinator != nil) {
65         return _persistentStoreCoordinator;
66     }
67     //Aqui estamos definindo o nome do arquivo de banco de dados -> ContatosCoreData.sqlite
68     NSURL *storeURL = [[self applicationDocumentsDirectory] URLByAppendingPathComponent:@"ContatosCoreData.sqlite"];
69
70     NSError *error = nil;
71     _persistentStoreCoordinator = [[NSPersistentStoreCoordinator alloc] initWithManagedObjectModel:[self managedObjectModel]];
72     if (![_persistentStoreCoordinator addPersistentStoreWithType:NSSQLiteStoreType configuration:nil URL:storeURL options:nil
73             error:&error]) {
74
75         NSLog(@"Unresolved error %@", error, [error userInfo]);
76         abort();
77     }
78
79     return _persistentStoreCoordinator;
80 }
```

Core Data

- No AppDelegate.m, continue a programação.

```
103 #pragma mark Application's directory
104
105 /**
106  * Returns the URL to the application's Documents directory.
107  */
108 - (NSURL *)applicationDocumentsDirectory {
109     NSURL *path;
110     path = [[[NSFileManager defaultManager] URLsForDirectory:NSDocumentDirectory inDomains:NSUserDomainMask] lastObject];
111     NSLog(@"DB PATH %@", path);
112     return path;
113 }
114 }
```

Core Data

Reforçando

- Todo esse código é gerado pelo Wizard do Xcode quando criamos um projeto: New-> Project -> iOS -> Application -> escolhemos um dos templates e em seguida clicamos no checkbox: "Core Data".
- As duas coisas que normalmente podem ser mudadas são o nome do arquivo `xcdatamodeld` para Model na página 38, e o nome do arquivo de banco de dados para `ContatosCoreData.sqlite` nas páginas 39 e 40, o restante é padrão.

Core Data

Programando o Controller para gravar os dados e
o Controller para Exibir em uma lista - Parte 4

X-Code

Prof. Agesandro Scarpioni

Core Data

- Em contatosViewController.h., inclua uma linha de #import da classe AppDelegate.h(1) e um propriedade getset do tipo NSManagedObject para contatos(2).

```
1 //  
2 // ContatosViewController.h  
3 // CoreData_ExemploUso  
4 //  
5 // Created by agesandro scarpioni on 22/06/14.  
6 // Copyright (c) 2014 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 #import <UIKit/UIKit.h>  
10 #import "AppDelegate.h"  
11  
12 @interface ContatosViewController : UIViewController  
13 @property (weak, nonatomic) IBOutlet UITextField *txtNome;  
14 @property (weak, nonatomic) IBOutlet UITextField *txtTelefone;  
15 @property (weak, nonatomic) IBOutlet UITextField *txtEmail;  
16 @property (strong) NSManagedObject *contatos;  
17  
18  
19  
20 @end  
21
```



Core Data

FIAF

- Programe as linhas abaixo no ContatosViewController.m do IBAction "salvar", não esqueça do @synthesize(1) antes do método. Já é possível testar a gravação, só não é possível ler os dados salvos, execute com command + R.

```
1 //  
2 // ContatosViewController.m  
3 // CoreData_ExemploUso  
4 //  
5 // Created by agesandro scarpioni on 22/06/14.  
6 // Copyright (c) 2014 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 #import "ContatosViewController.h"  
10  
11 @interface ContatosViewController ()  
12 @end 1  
13  
14 @implementation ContatosViewController  
15 @synthesize txtEmail,txtNome,txtTelefone;  
16 @synthesize contatos;  
17  
18  
19 - (IBAction)salvar:(id)sender {  
20     //AppDelegate da aplicação  
21     AppDelegate *appDelegate = (AppDelegate *) [[UIApplication sharedApplication] delegate];  
22     //Context para salvar/deletar/buscar objetos  
23     NSManagedObjectContext *context = [appDelegate managedObjectContext];  
24  
25     //Cria uma instância do contato (inserindo no manage object context)  
26     NSManagedObject *Novocontato = [NSEntityDescription insertNewObjectForEntityForName:@"Contato" inManagedObjectContext:context];  
27     [Novocontato setValue:txtNome.text forKey:@"nome"];  
28     [Novocontato setValue:txtTelefone.text forKey:@"telefone"];  
29     [Novocontato setValue:txtEmail.text forKey:@"email"];  
30     NSError *error;  
31     //Salvando contexto  
32     [context save:&error];  
33  
34     [self.navigationController popViewControllerAnimated:YES];  
35 }
```

Obs: Caso queira pode melhorar o save, adicionando o IF no destaque, você também pode substituir o NSLog para um UIAlertView.

Core Data

- Programe a linha de @Property do tipo Array mutável (1) em `ContatosTableViewController.h` e o respectivo @synthesize em `ContatosTableViewController.m`, além do import de "AppDelegate" (2).

```
1 //  
2 // ContatosTableViewController.h  
3 // CoreData_ExemploUso  
4 //  
5 // Created by agesandro scarpioni on 22/06/14.  
6 // Copyright (c) 2014 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 #import <UIKit/UIKit.h>  
10  
11  
12 @interface ContatosTableViewController : UITableViewController  
13 @property (strong) NSMutableArray *contatos;  
14  
15  
16 @end
```



```
1 //  
2 // ContatosTableViewController.m  
3 // CoreData_ExemploUso  
4 //  
5 // Created by agesandro scarpioni on 22/06/14.  
6 // Copyright (c) 2014 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 #import "ContatosTableViewController.h"  
10 #import "AppDelegate.h"  
11  
12 @interface ContatosTableViewController ()  
13  
14 @end  
15  
16 @implementation ContatosTableViewController  
17 @synthesize contatos;
```

Core Data

- Programe em `ContatosTableViewController.m` as linhas necessárias para exibir os dados no `tableView` ordenados pelo nome.

```
34
35 -(void) viewDidAppear:(BOOL)animated{
36     [super viewDidAppear:YES];
37     //AppDelegate da aplicação
38     AppDelegate *appDelegate = (AppDelegate *) [[UIApplication sharedApplication] delegate];
39     //Context para salvar/deletar/buscar objetos
40     NSManagedObjectContext *context = [appDelegate managedObjectContext];
41     //cria o Request da consulta @Contato é a Entity em nosso Model.xcdatamodel
42     NSFetchedResultsController *request = [[NSFetchedResultsController alloc] initWithEntityName:@"Contato"];
43     //Ordena a consulta por nome
44     NSSortDescriptor *ordem = [[NSSortDescriptor alloc] initWithKey:@"nome" ascending:YES];
45     NSArray *ordenados = [NSArray arrayWithObject:ordem];
46     [request setSortDescriptors:ordenados];
47     //executa o request e armazena no array contatos.
48     contatos = [[context executeFetchRequest:request error:nil]mutableCopy];
49     [self.tableView reloadData];
50 }
51 }
```

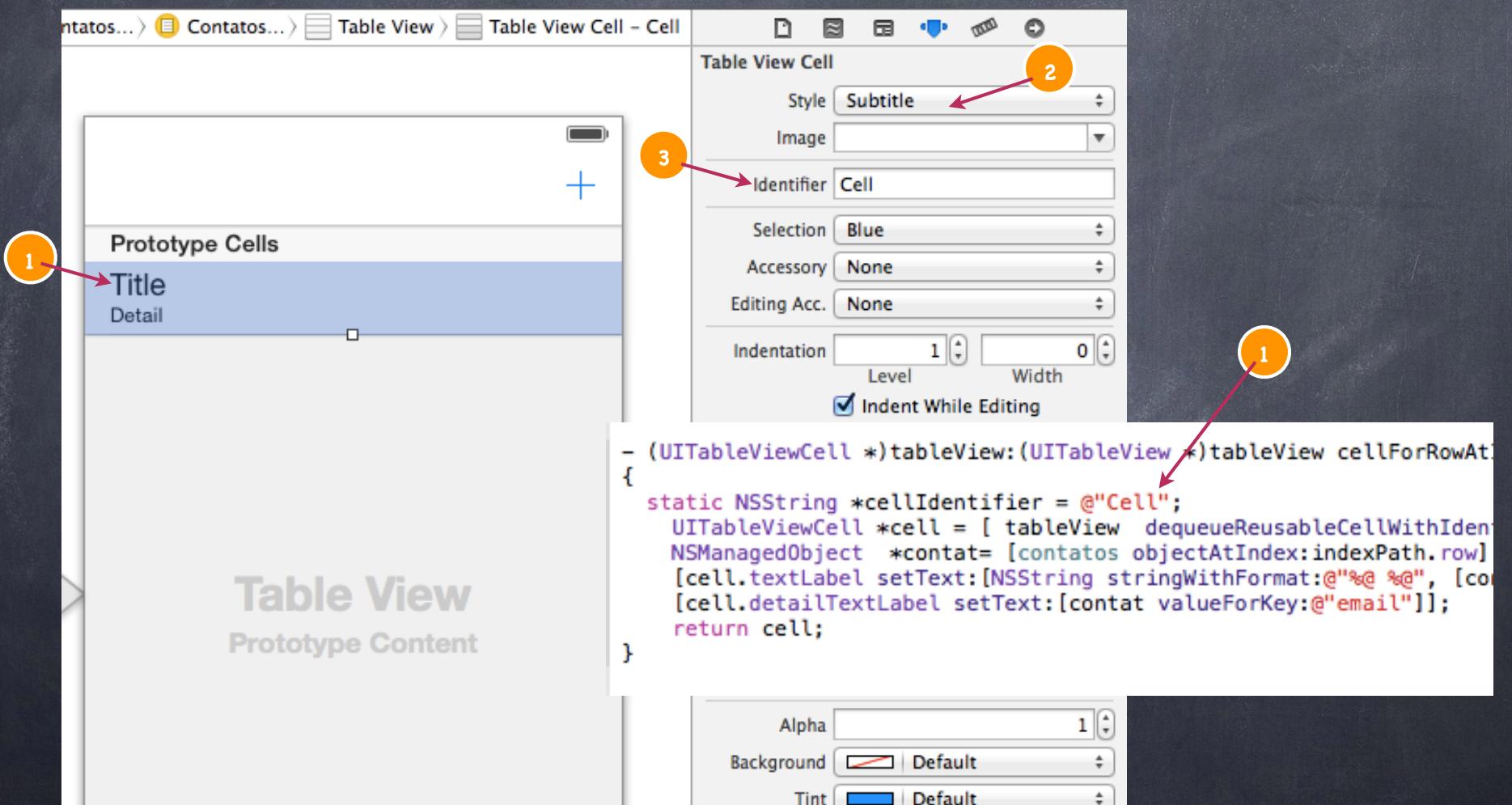
Core Data

- Programe em `ContatosTableViewController.m` as linhas necessárias para gerar as células do Tableview, as assinaturas dos métodos já estão disponíveis na classe que criamos.

```
57
58 #pragma mark - Table view data source
59
60 - (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
61 {
62     // Return the number of sections.
63     return 1;
64 }
65
66 - (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
67 {
68     // Return the number of rows in the section.
69     return contatos.count;
70 }
71
72 - (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
73 {
74     static NSString *cellIdentifier = @"Cell";
75     UITableViewCell *cell = [ tableView dequeueReusableCellWithIdentifier:cellIdentifier forIndexPath:indexPath];
76     NSManagedObject *contat= [contatos objectAtIndex:indexPath.row];
77     [cell.textLabel setText:[NSString stringWithFormat:@"%@ %@", [contat valueForKey:@"nome"], [contat valueForKey:@"telefone"]]];
78     [cell.detailTextLabel setText:[contat valueForKey:@"email"]];
79     return cell;
80 }
81 }
```

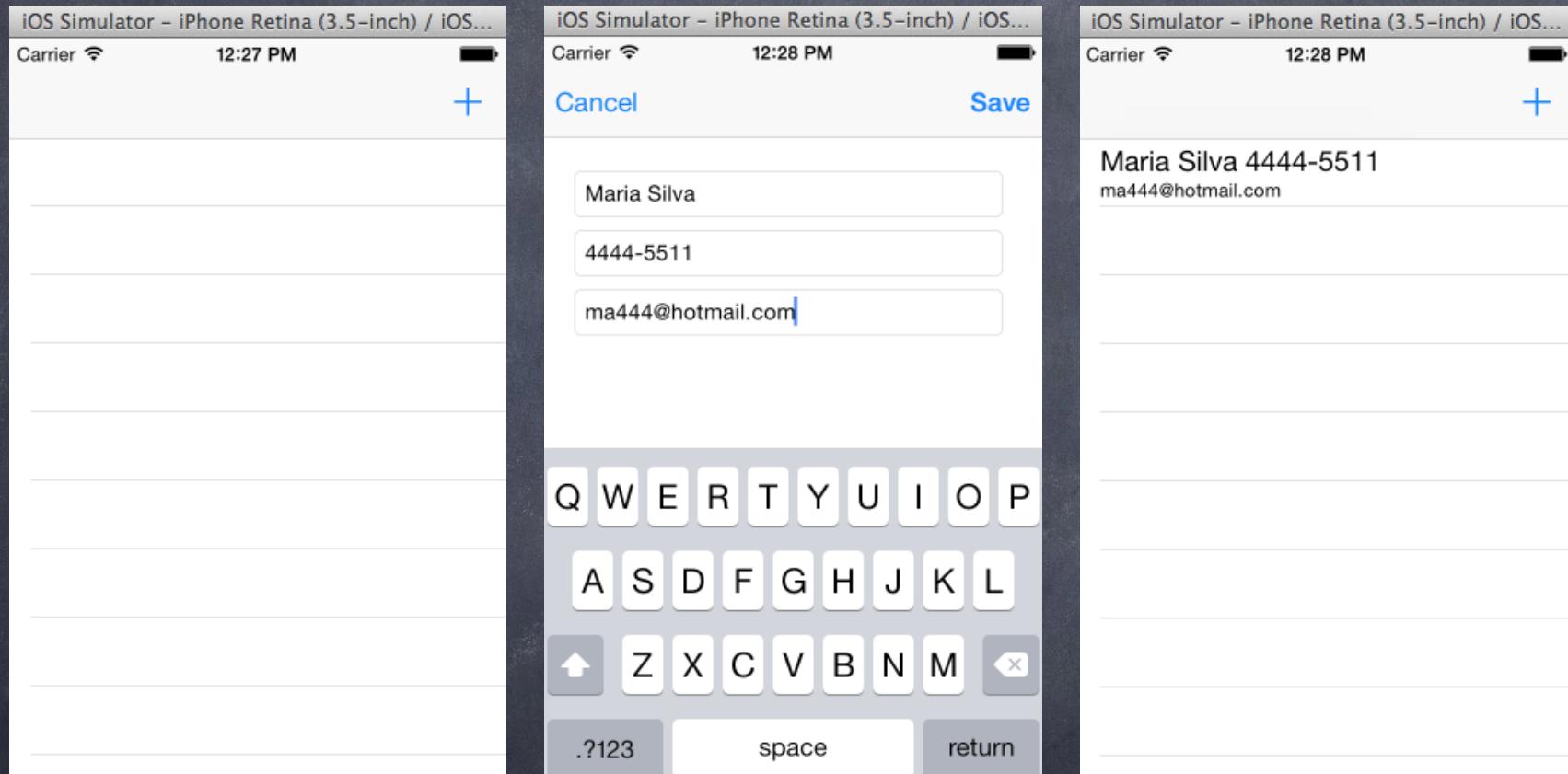
Core Data

- Selecione a célula do tableView (1), em Style escolha Subtitle (2), em Identifier informe Cell (3) que é o nome informado na programação da página anterior (4).



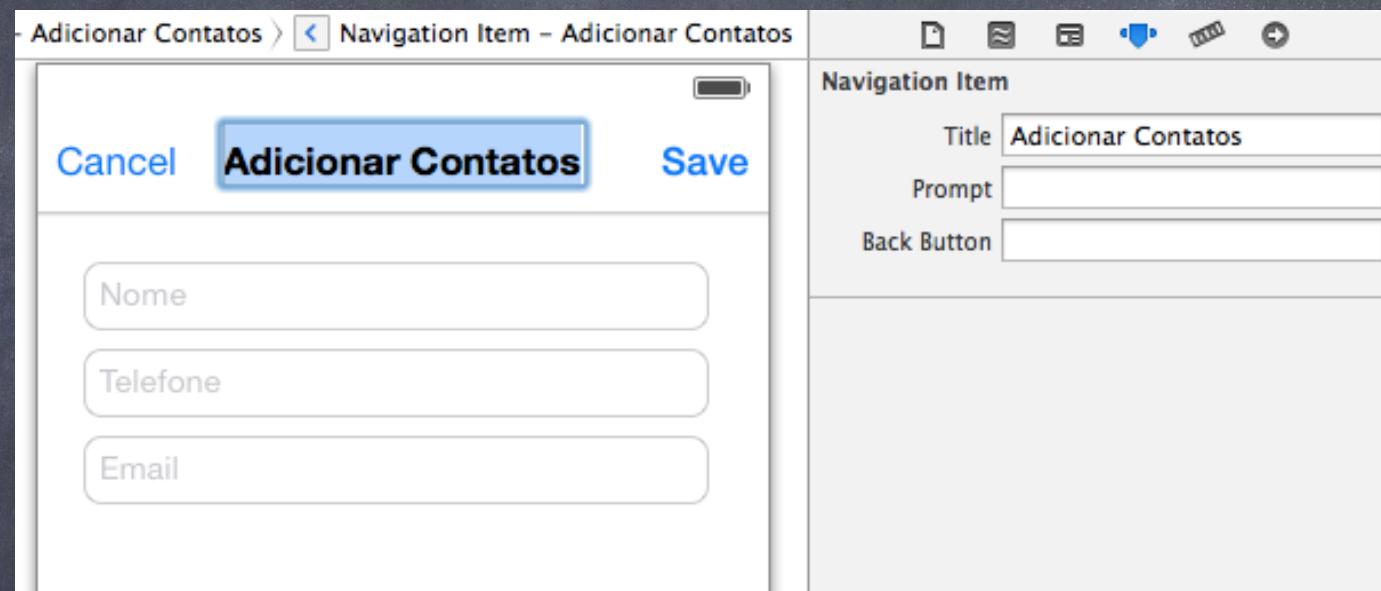
Core Data

• Command + R, faça os testes para a inclusão.



Core Data

- Clique duas vezes no centro da barra e digite: "Adicionar Contatos".



Core Data

Programando o Controller para apagar ou alterar os dados - Parte 5

X-Code

Prof. Agesandro Scarpioni

Core Data

- Localize o método `commitEditingStyle` (2) e retire a marcação de comentário(3), esse bloco possui a lógica para apagar os dados do TableView, veja a programação no próximo slide.

The screenshot shows the Xcode interface with the project 'CoreData_ExemploUso' selected. The 'ContatosTableViewController.m' file is open in the editor. Three annotations are present:

- Annotation 1:** A red arrow points from the left margin to the line number 71, which contains the code for `cellForRowAtIndexPath`.
- Annotation 2:** A red circle highlights the line number 93, which contains the code for `commitEditingStyle`.
- Annotation 3:** A red arrow points from the left margin to the line number 102, which contains the closing brace for the `commitEditingStyle` method.

```
71 - (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
72 {
73     static NSString *cellIdentifier = @"Cell";
74     UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:cellIdentifier forIndexPath];
75     ;
76     NSManagedObject *contat = [contatos objectAtIndex:indexPath.row];
77     [cell.textLabel setText:[NSString stringWithFormat:@"%@ %@", [contat valueForKey:@"nome"], [contat
78         valueForKey:@"telefone"]]];
79     [cell.detailTextLabel setText:[contat valueForKey:@"email"]];
80     return cell;
81 }
82
83
84 // Override to support conditional editing of the table view.
85 - (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath *)indexPath
86 {
87     // Return NO if you do not want the specified item to be editable.
88     return YES;
89 }
90
91 /*
92 // Override to support editing the table view.
93 - (void)tableView:(UITableView *)tableView commitEditingStyle:(UITableViewCellEditingStyle)editingStyle
94     forRowAtIndexPath:(NSIndexPath *)indexPath
95 {
96     if (editingStyle == UITableViewCellEditingStyleDelete) {
97         // Delete the row from the data source
98         [tableView deleteRowsAtIndexPaths:@[indexPath] withRowAnimation:UITableViewRowAnimationFade];
99     } else if (editingStyle == UITableViewCellEditingStyleInsert) {
100        // Create a new instance of the appropriate class, insert it into the array, and add a new row to the
101        // table view
102    }
103 }
```

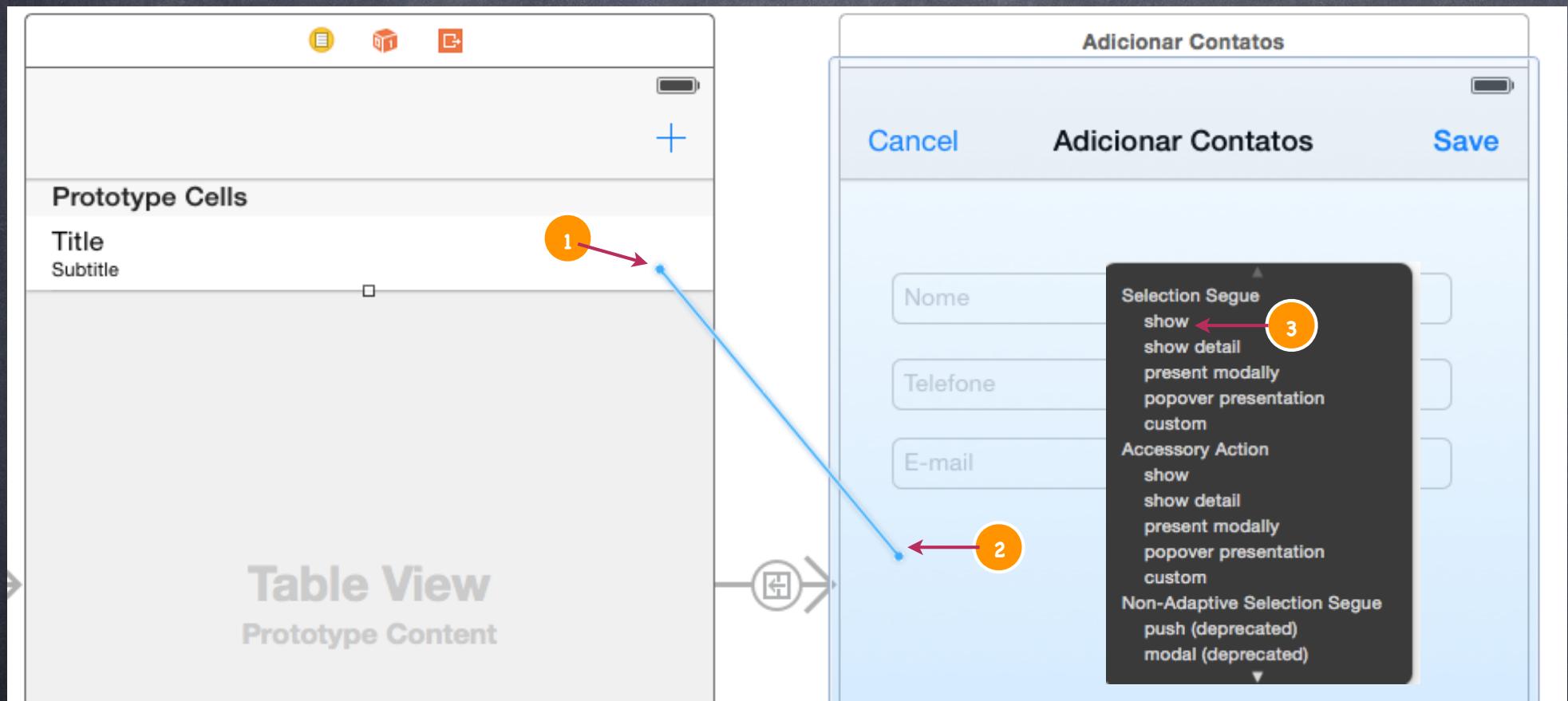
Core Data

- Para excluir um contato, digite as linhas 96 até a 99 para preparar o context, complete o if com as linhas 102 até a 111, o restante já pertencia ao código que estava comentado.

```
91 // Override to support editing the table view.
92 - (void)tableView:(UITableView *)tableView commitEditingStyle:(UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:(NSIndexPath *)indexPath
93 {
94
95     //AppDelegate da aplicação
96     AppDelegate *appDelegate = (AppDelegate *) [[UIApplication sharedApplication] delegate];
97     //Context para salvar/deletar/buscar objetos
98     NSManagedObjectContext *context = [appDelegate managedObjectContext];
99
100    if (editingStyle == UITableViewCellEditingStyleDelete) {
101        // Apaga o objeto do banco de dados
102        [context deleteObject:[contatos objectAtIndex:indexPath.row]];
103        // invoca o método "Save" para comitar a mudança
104        NSError *error;
105        if (![context save:&error]){
106            NSLog(@"Erro ao deletar %@ %@", error, [error localizedDescription]);
107            return;
108        }
109        //remove o contato do TableView e do array
110        [contatos removeObjectAtIndex:indexPath.row];
111        [tableView deleteRowsAtIndexPaths:@[indexPath] withRowAnimation:UITableViewRowAnimationFade];
112    } else if (editingStyle == UITableViewCellEditingStyleInsert) {
113        // Create a new instance of the appropriate class, insert it into the array, and add a new row
114        // to the table view
115    }
116}
117}
```

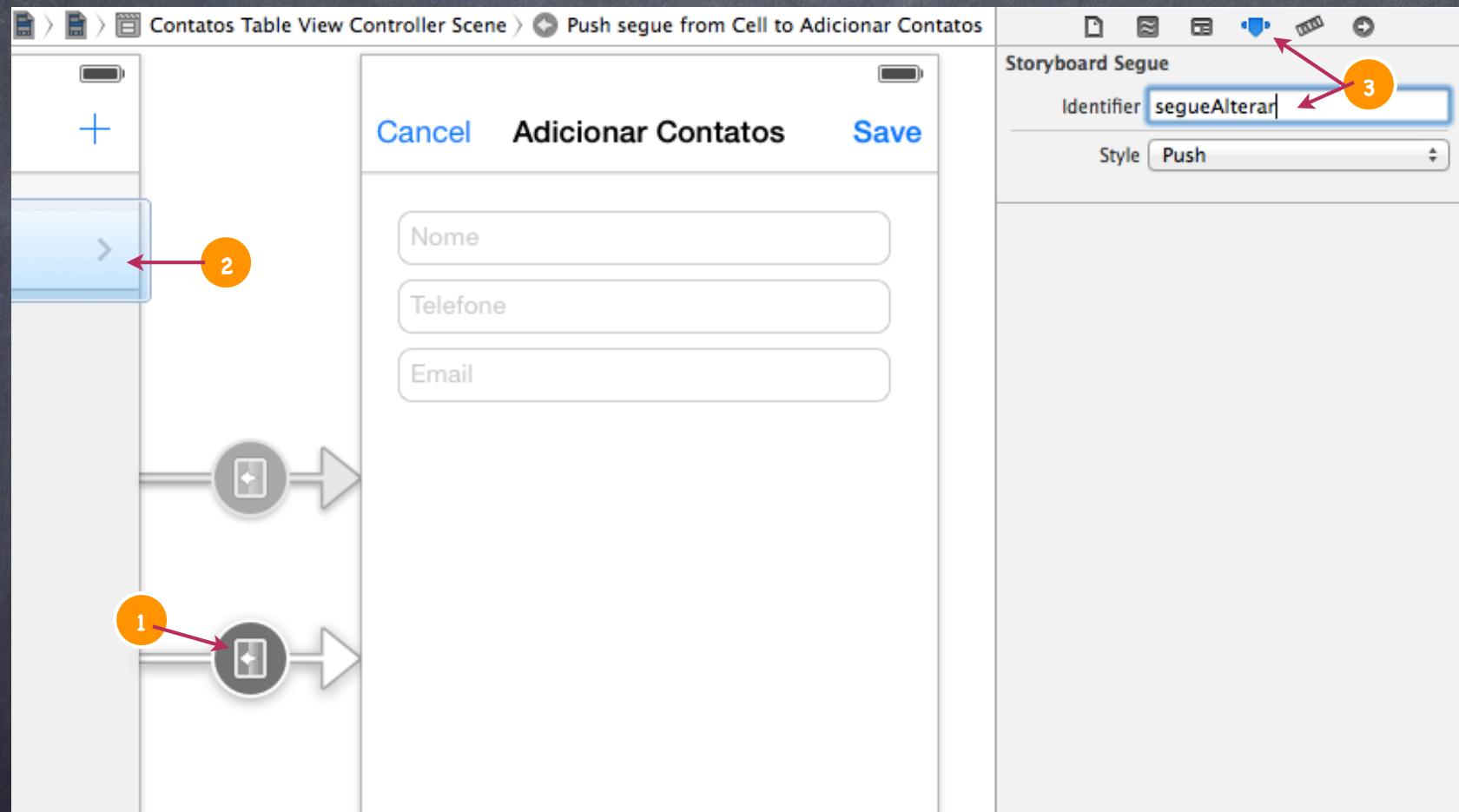
Core Data

- Preparar a Table para a edição dos dados, selecione a célula (1) e arraste até a UITableViewController (2), escolha Show (3).



Core Data

- Selecione a Segue que você acabou de criar (1), observe se a Segue é referente a célula(2), altere a propriedade Identifier para: "segueAlterar"(3).



Core Data

- Em contatosViewController.m adicione o "if else" da linha 25 até a 32 para diferenciarmos o update de um insert.

```
18
19 - (IBAction)salvar:(id)sender {
20     //AppDelegate da aplicação
21     AppDelegate *appDelegate = (AppDelegate *) [[UIApplication sharedApplication] delegate];
22     //Context para salvar/deletar/buscar objetos
23     NSManagedObjectContext *context = [appDelegate managedObjectContext];
24
25     if (contatos){
26         //atualizando um carro existente
27         [contatos setValue:txtNome.text forKey:@"nome"];
28         [contatos setValue:txtTelefone.text forKey:@"telefone"];
29         [contatos setValue:txtEmail.text forKey:@"email"];
30     }else{
31         //adicionando um carro novo
32         //Cria uma instância do contato (inserindo no manage object context)
33         NSManagedObject *Novocontato = [NSEntityDescription insertNewObjectForEntityForName:@"Contato"
34                                         inManagedObjectContext:context];
35         [Novocontato setValue:txtNome.text forKey:@"nome"];
36         [Novocontato setValue:txtTelefone.text forKey:@"telefone"];
37         [Novocontato setValue:txtEmail.text forKey:@"email"];
38     }
39     NSError *error;
40     //Salvando contexto
41     if (![context save:&error]){
42         NSLog(@"Erro ao salvar %@ %@", error, [error localizedDescription]);
43     }
44
45     [self.navigationController popViewControllerAnimated:YES];
46 }
```

Core Data

- Para preparar o update, vá em contatosTableViewController.h e dê o import de contatosViewController.h.

```
1 //  
2 // ContatosTableViewController.h  
3 // CoreData_ExemploUso  
4 //  
5 // Created by agesandro scarpioni on 22/06/14.  
6 // Copyright (c) 2014 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 #import <UIKit/UIKit.h>  
10 #import "ContatosViewController.h"  
11  
12  
13 @interface ContatosTableViewController : UITableViewController  
14 @property (strong) NSMutableArray *contatos;  
15  
16  
17 @end  
18
```

Core Data

- Vá em `contatosTableViewController.m` e programe a passagem dos dados do `contatosTableViewController` para o `contatosViewController` (1), faça o import de `AppDelegate` (2).

```
1 //  
2 // ContatosTableViewController.m  
3 // CoreData_ExemploUso  
4 //  
5 // Created by agesandro scarpioni on 22/06/14.  
6 // Copyright (c) 2014 Agesandro Scarpioni. All rights reserved.  
7 //  
8 // 2  
9 #import "ContatosTableViewController.h"  
10 #import "AppDelegate.h"  
11  
12 @interface ContatosTableViewController : UITableViewController  
13  
14 @end  
15  
16 @implementation ContatosTableViewController  
17 @synthesize contatos;  
18  
19 -(void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender{  
20     if ([[segue identifier] isEqualToString:@"segueAlterar"]){  
21         // 1  
22         NSManagedObject *selectedContato = [contatos objectAtIndex:[[self.tableView indexPathForSelectedRow] row]];  
23         ContatosViewController *destinoViewController = segue.destinationViewController;  
24         destinoViewController.contatos = selectedContato;  
25     }  
26 }
```

Core Data

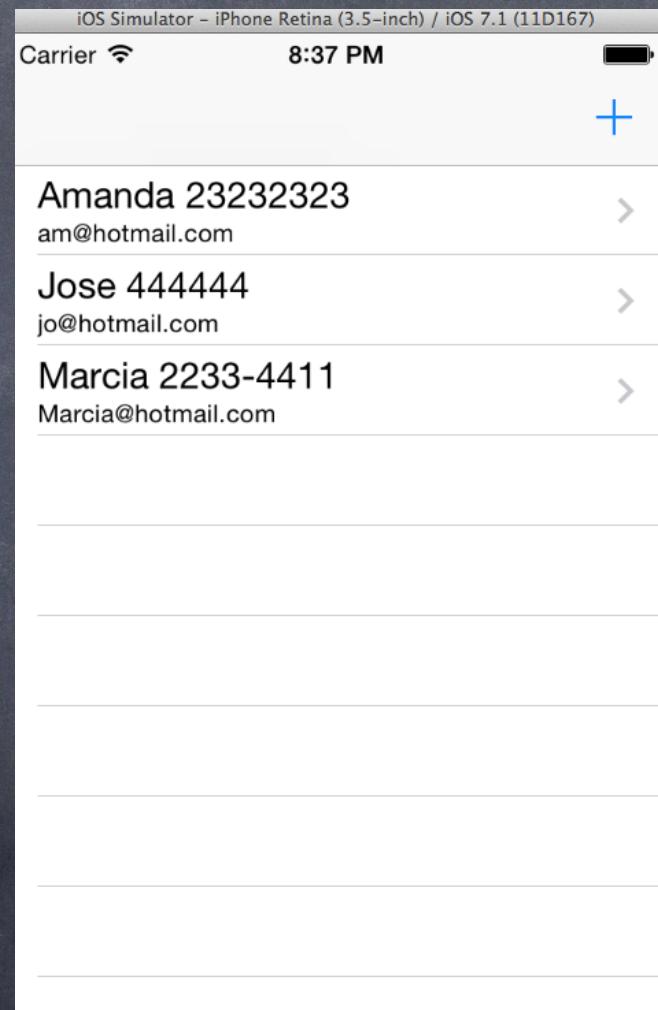
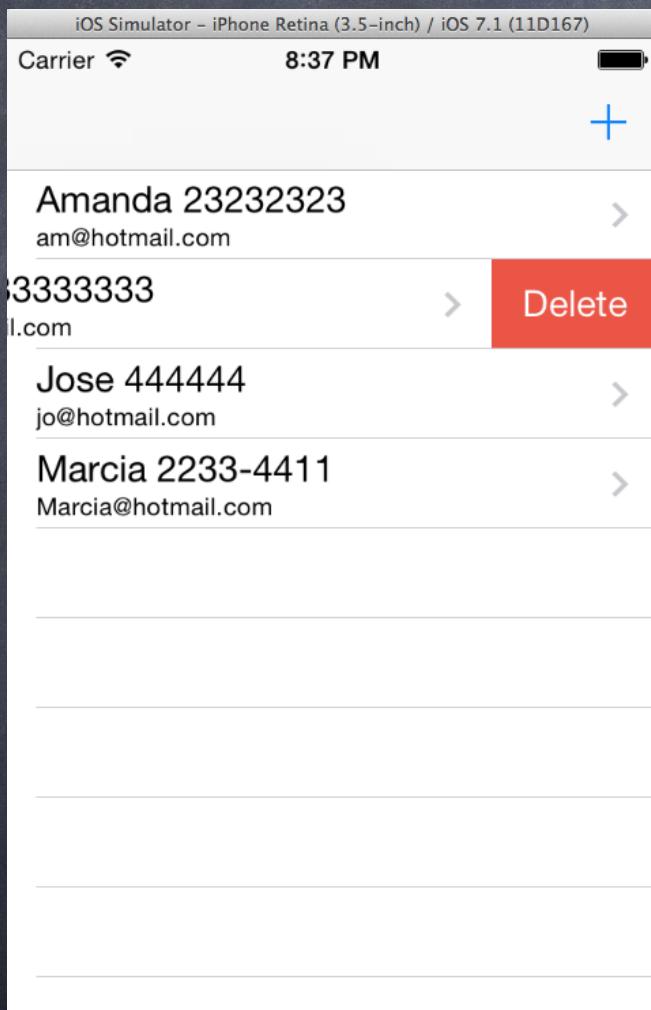
- Em contatosViewController.m, faça a programação do IF para exibir os dados da passagem da Table para essa View.

```
59
60 - (void)viewDidLoad
61 {
62     [super viewDidLoad];
63     if (contatos){
64         [txtNome setText:[contatos valueForKey:@"nome"]];
65         [txtTelefone setText:[contatos valueForKey:@"telefone"]];
66         [txtEmail setText:[contatos valueForKey:@"email"]];
67     }
68 }
69 }
70 }
```

Obs: O nome segueAlterar tem que ser exatamente igual ao nome dado no identifier da segue que criamos entre as duas views na pág. 56.

Core Data

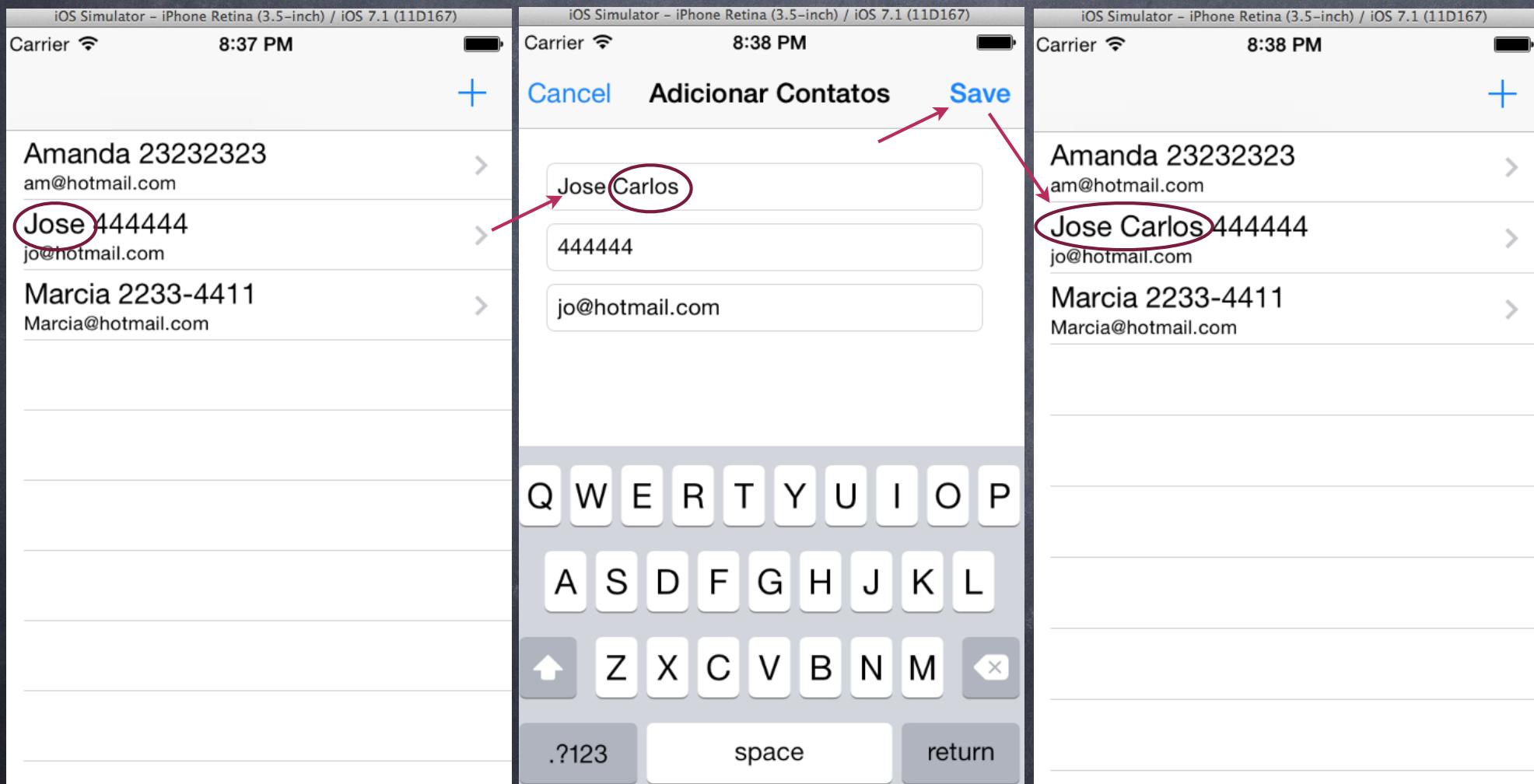
Execute o app com command + R para testar a exclusão.



Core Data

FIAP

• Agora vamos testar a edição



Prática

- ➊ Tente criar uma nova tela para armazenar outros dados como por exemplo: Dados de Jogadores de Futebol como nome, camisa, posição e time.