FIAP

Core Data (CRUD)

Persistência - Interface - Parte 1

X-Code com Swift
Prof. Agesandro Scarpioni
agesandro@fiap.com.br

Core Data

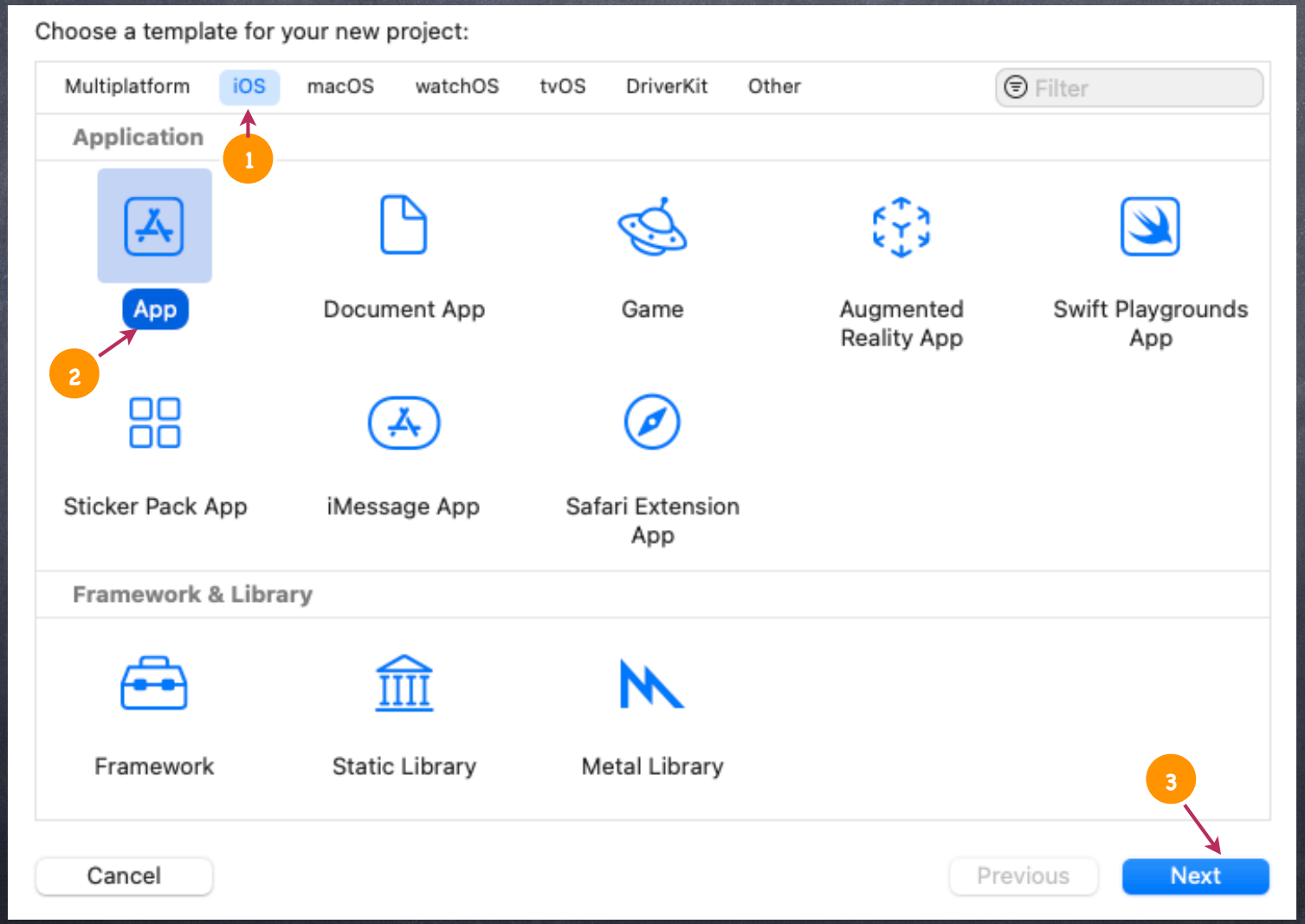
- Core Data é um framework de persistência fornecido pela Apple nos sistemas operacionais macOS e iOS. Ele permite que os dados organizados pelo modelo relacional de entidade-atributo sejam serializados em armazenamentos XML, binários ou SQLite.
- Existem diversas formas de persistir dados em uma aplicação, mas a mais robusta é usando banco de dados.
- Core Data é utilizado para trabalhar a camada Model da sua aplicação ele nos permite persistir dados em nosso aplicativo, utilizando banco de dados, de uma forma orientada a objetos.

Core Data

- Core Data não é um banco de dados, mas sim um gerenciador de grafoS que também inclui persistência. Trabalha (agrupa, filtra e organiza) os dados em memória.
- Nesta aula você irá persistir os dados e ver como é fácil navegar entre telas utilizando o storyboard com Navigation Controller e TableView.

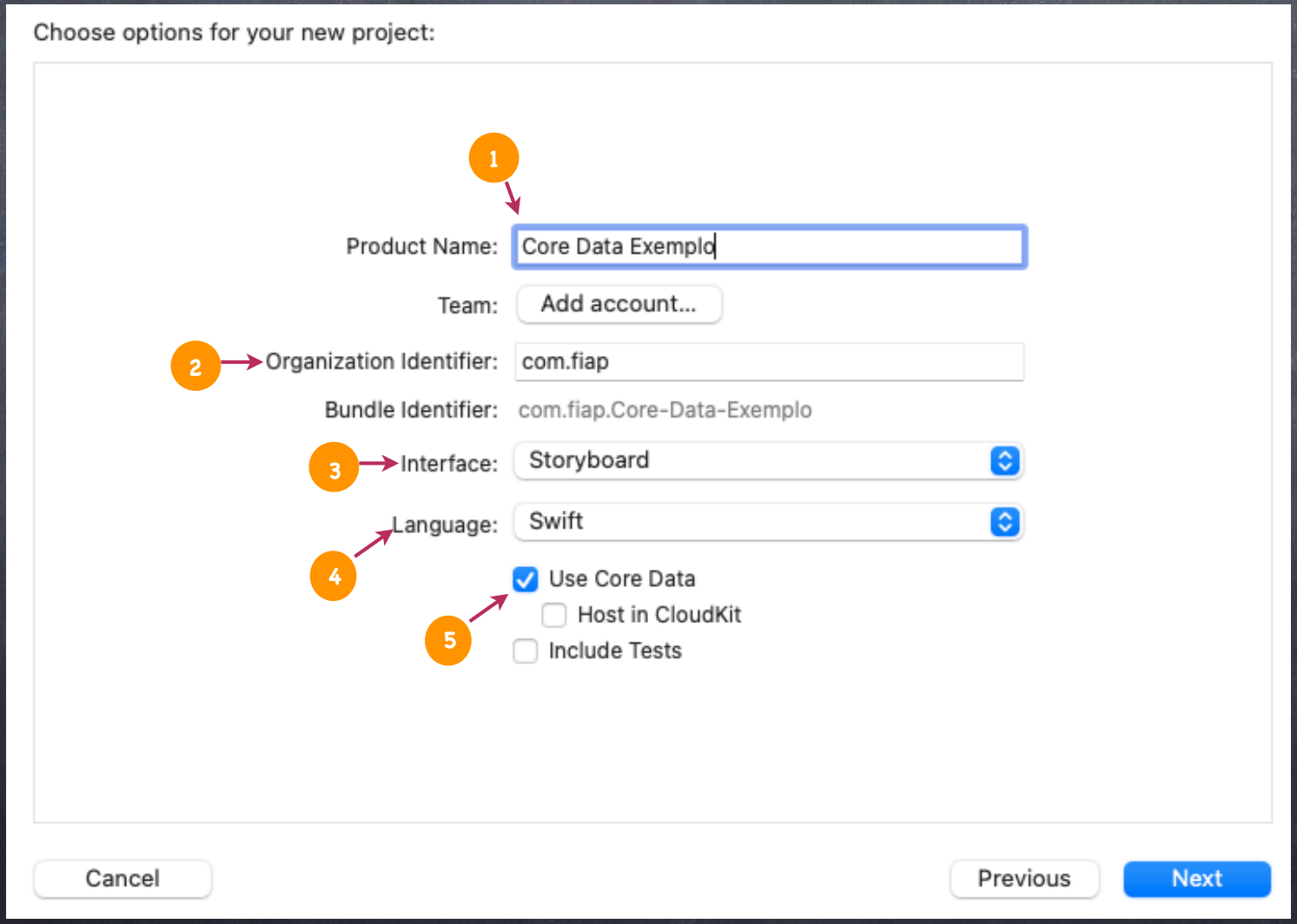
Template

- Vamos criar um projeto novo do tipo IOS (1), App (2) e clique em Next(3).



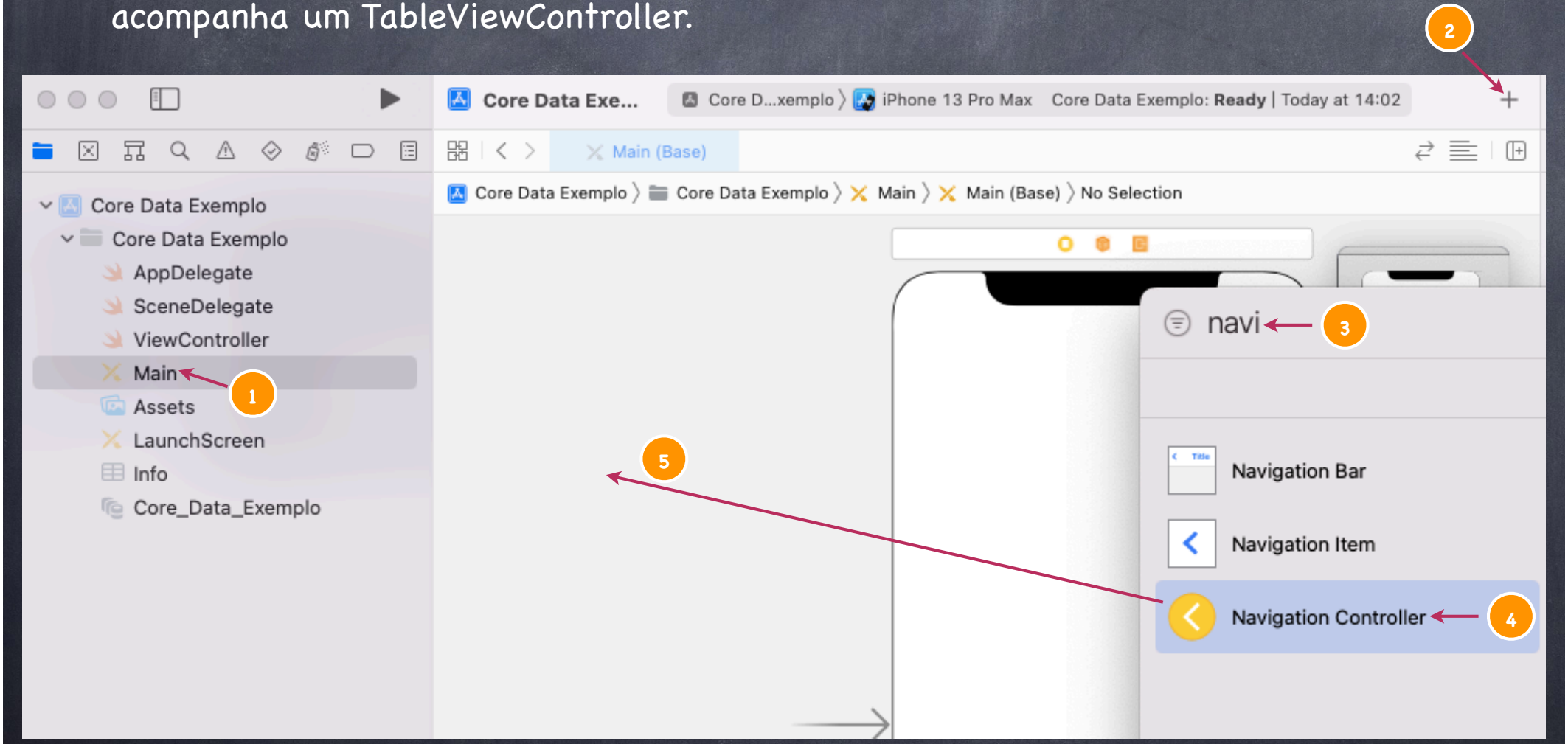
Criando o Projeto

- Nomeie o projeto como "Core Data Exemplo" (1), defina uma URL invertida em Organization Identifier (2), selecione Storyboard em User Interface (3), language: Swift (4), marque o checkbox: Use Core Data(5). IMPORTANTE: Para projetos Core Data não use "-" no nome do arquivo.



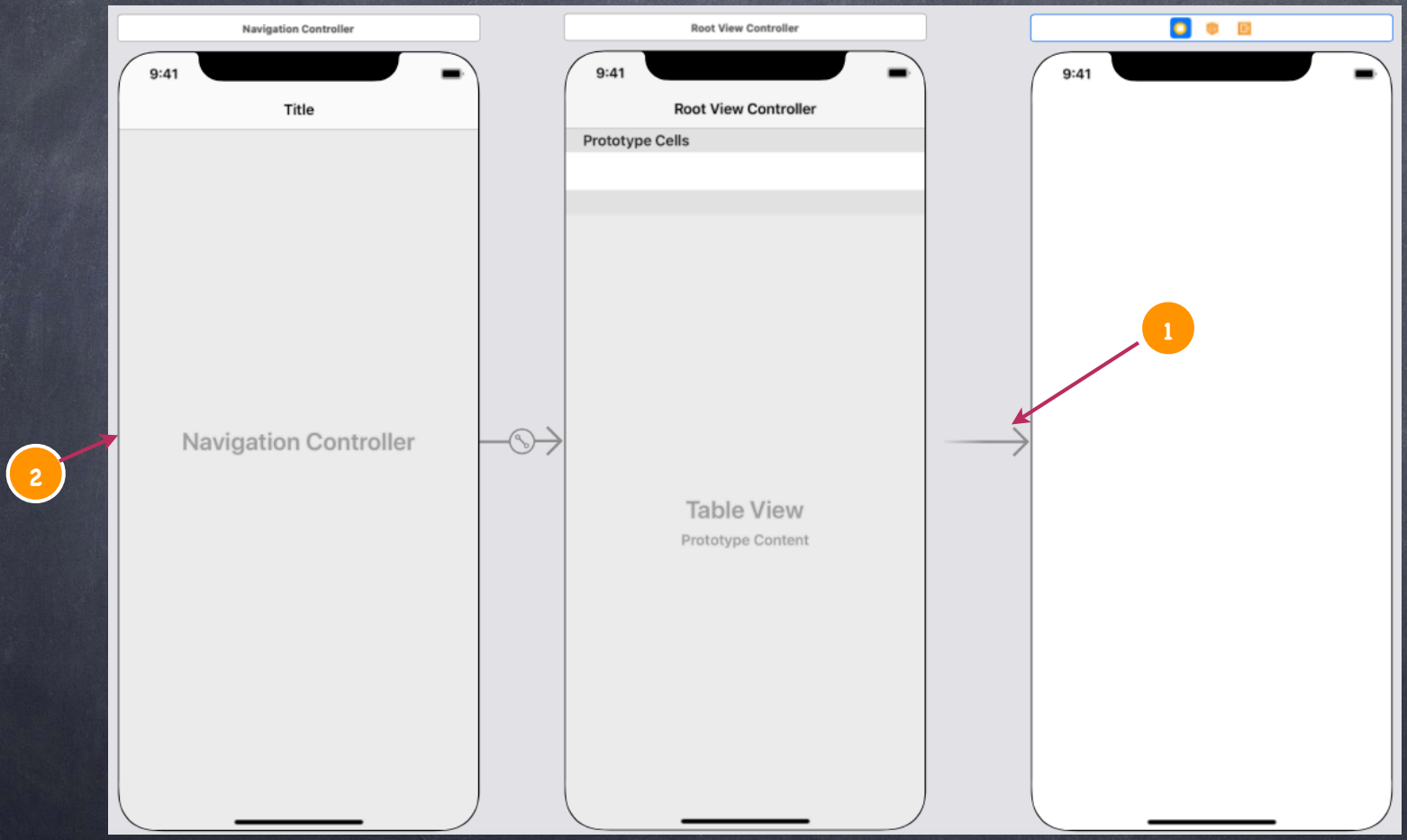
Navigation Controller

- No seu Main.storyboard (1) inclua um Navigation Controller (2,3,4,5), este objeto acompanha um TableViewController.



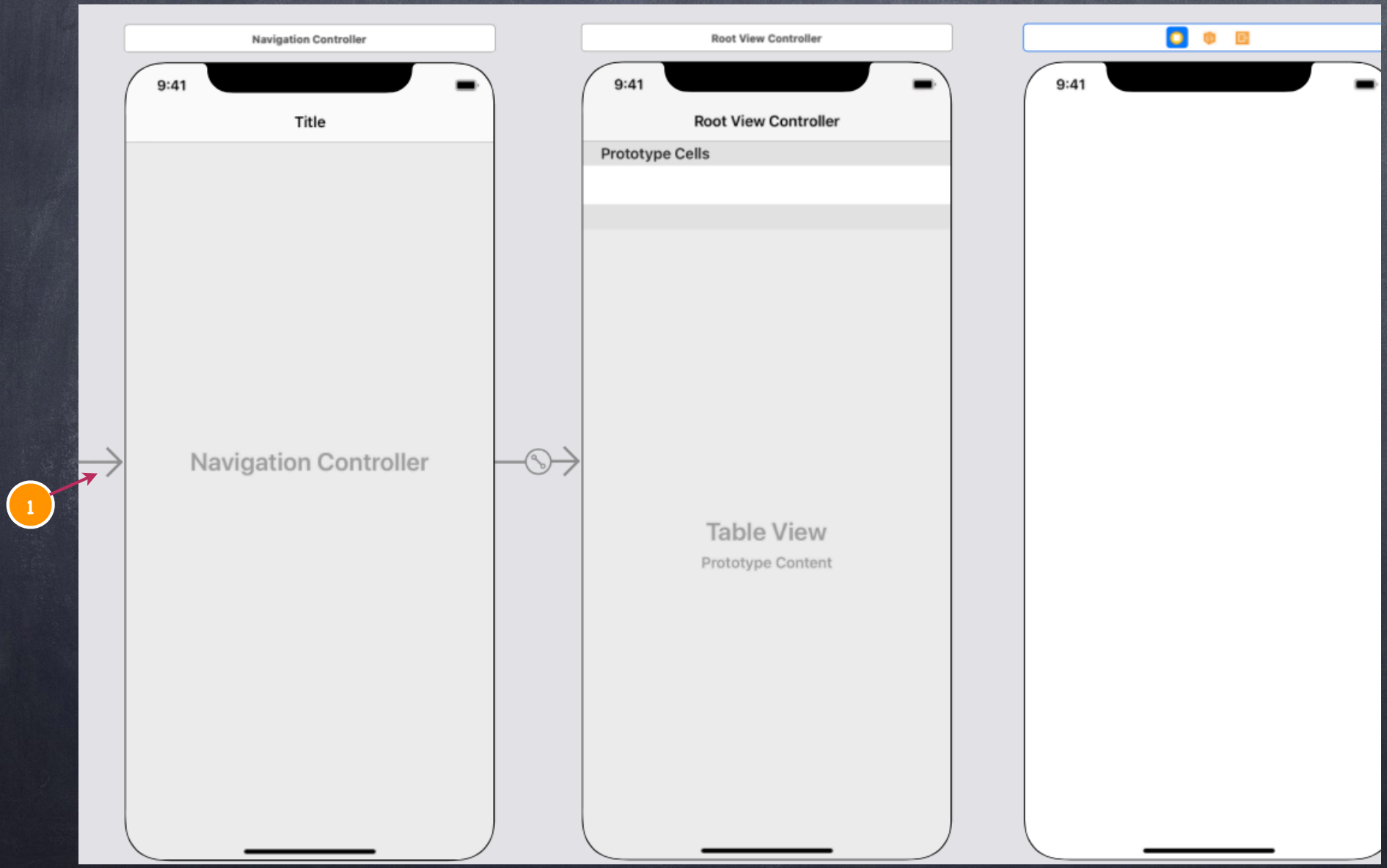
Initial View Controller

- Após incluir o Navigation Controller sua tela ficará parecida com o exemplo abaixo, arraste o "Initial View Controller" (1) para antes do "Navigation Controller" (2)



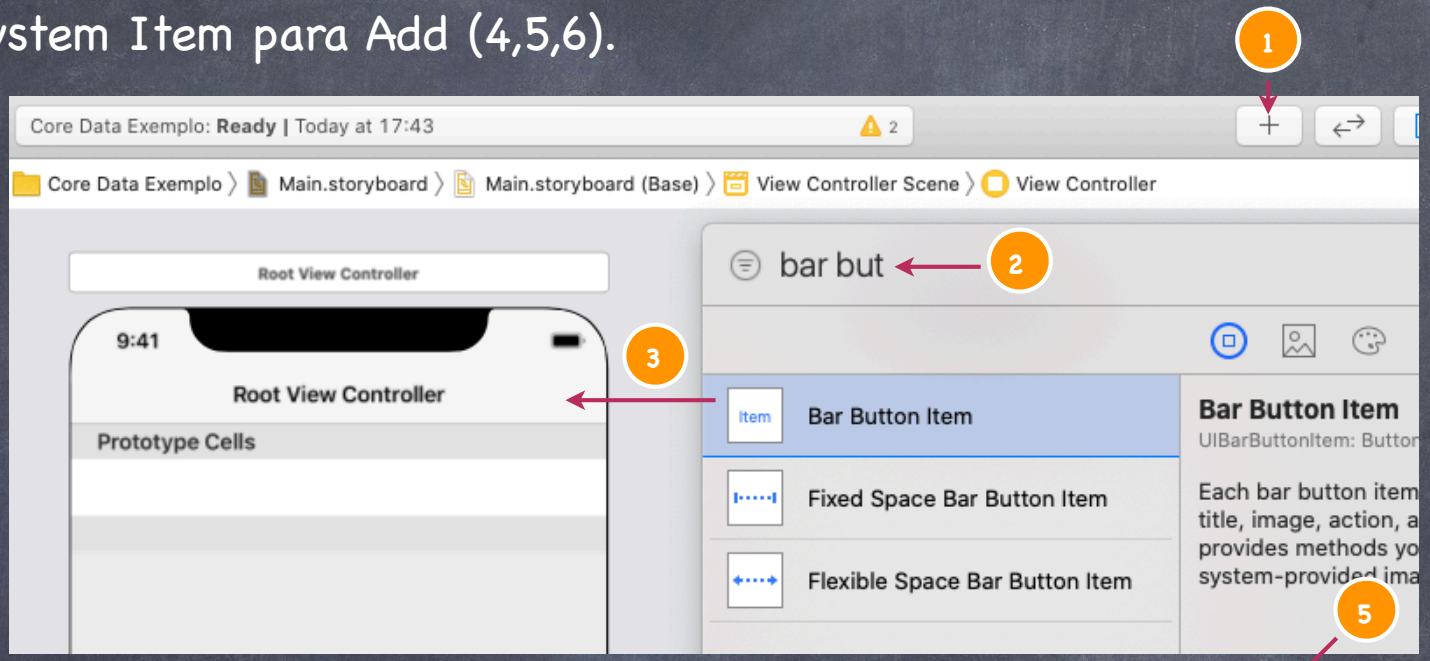
Initial View Controller

- Note que a seta "Initial View Controller" (1) mudou de local



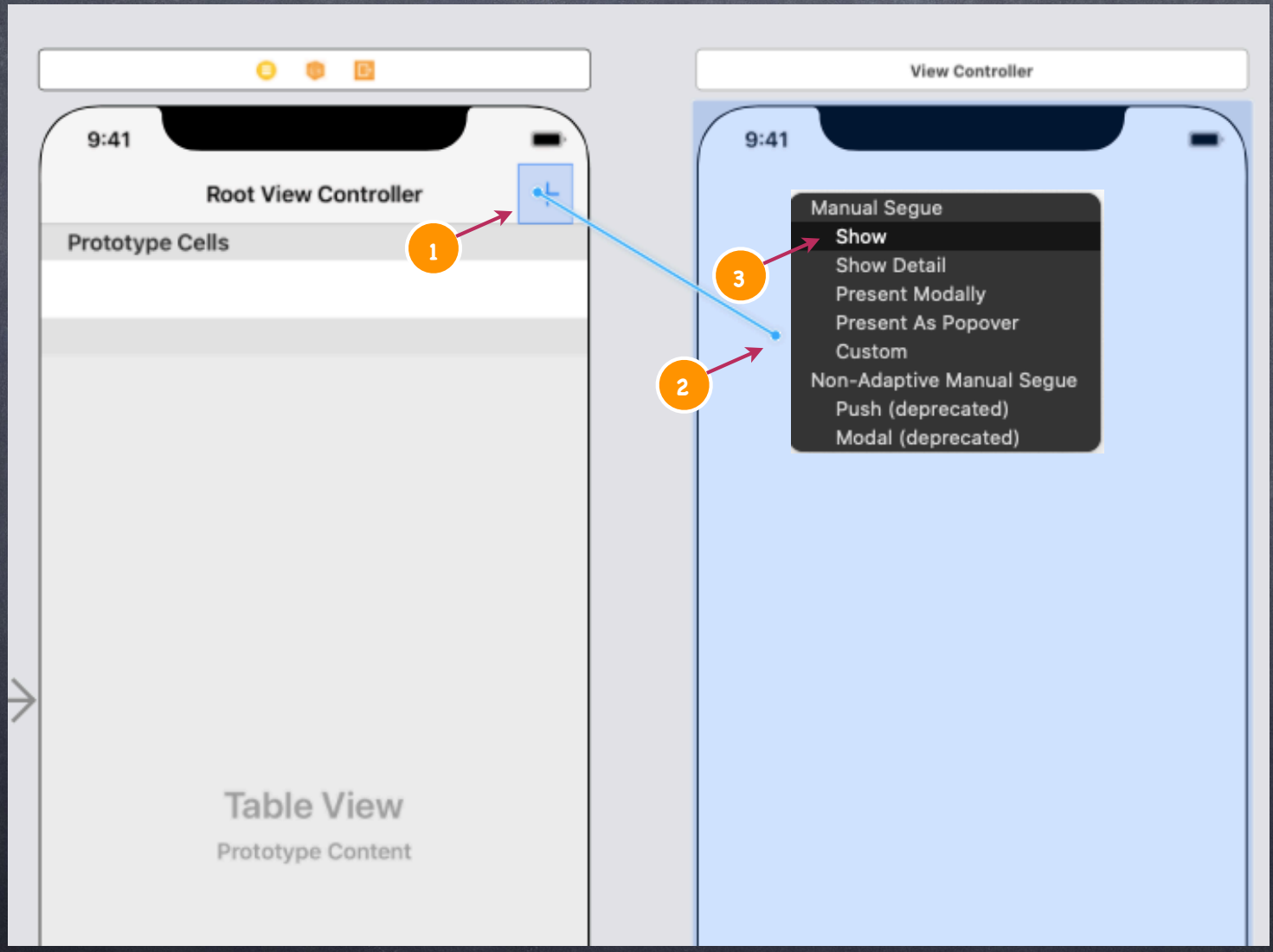
Bar Button

- Inclua na barra de navegação da TableView um objeto Bar Button Item (1,2,3), altere o atributo System Item para Add (4,5,6).



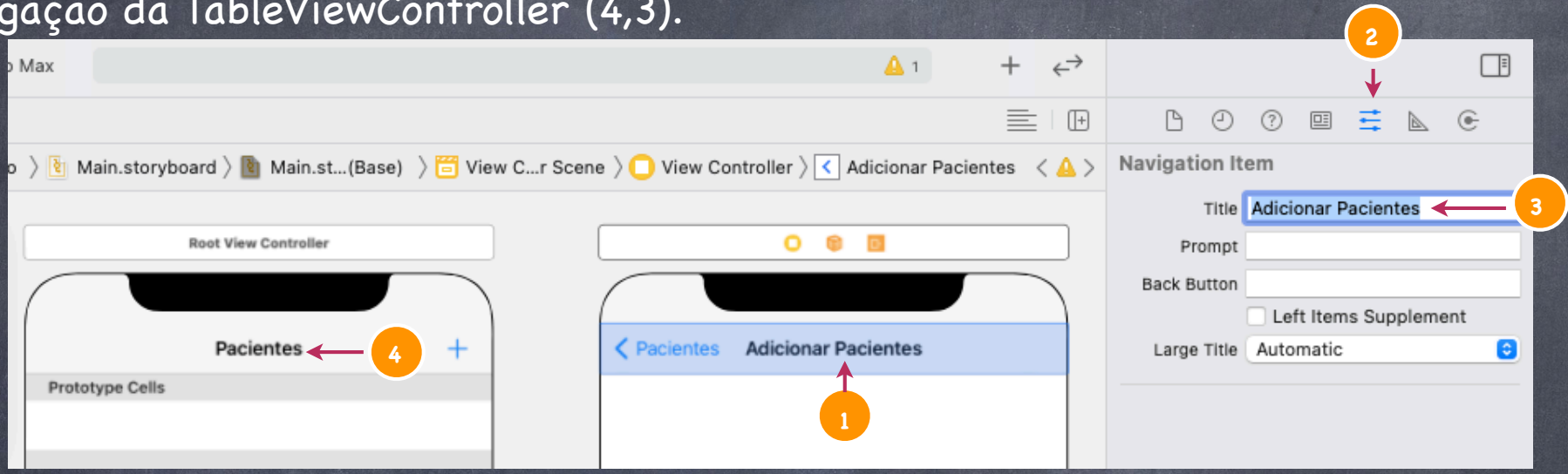
Add Bar Button

- Ligue o item Add(1) a ViewController(2) e escolha Show(3).

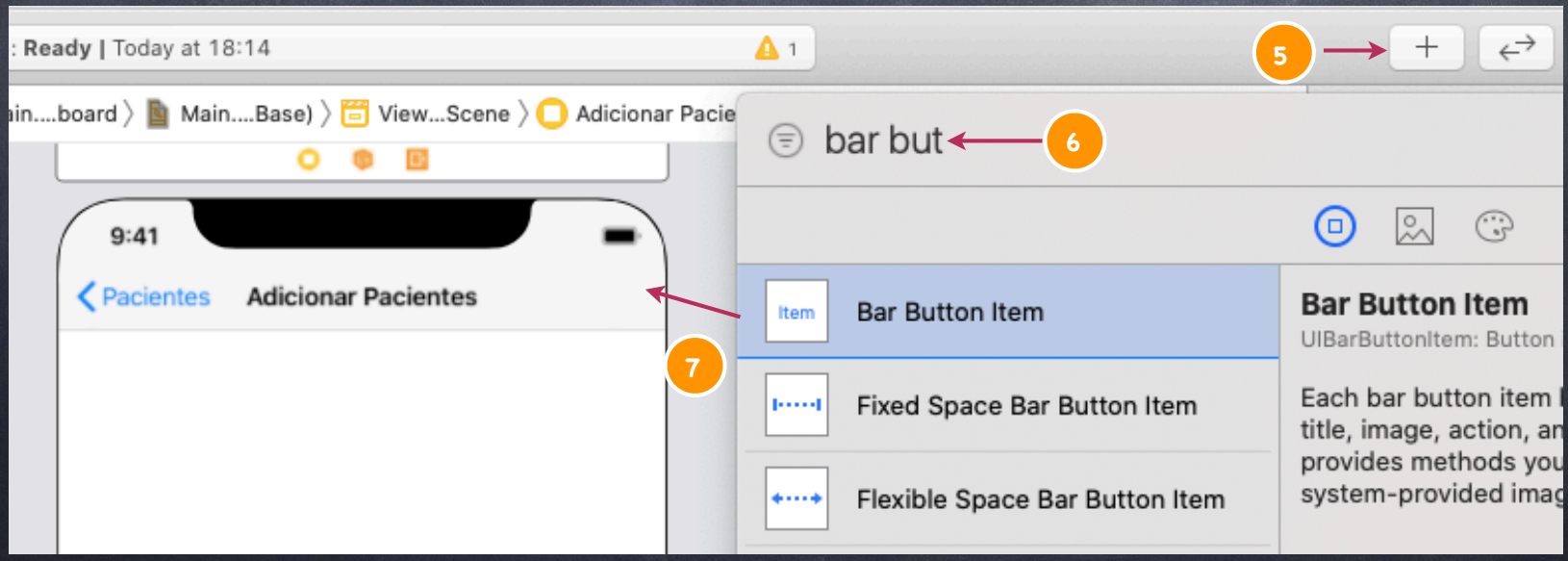


Save Bar Button

- Altere os títulos da barra de navegação da ViewController (1,2,3) e o título da barra de navegação da TableViewController (4,3).

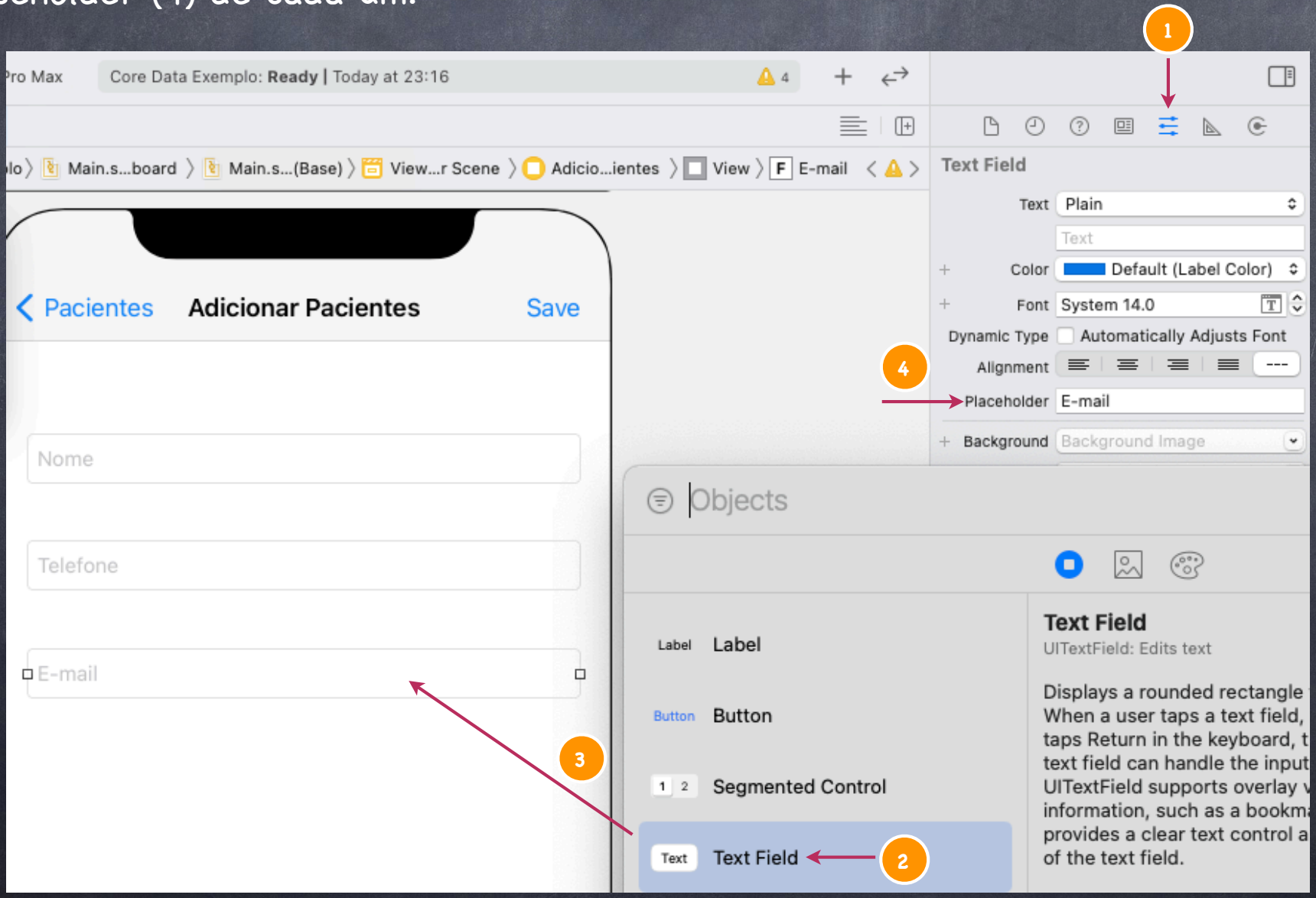


- Adicione um Bar Button Item (5, 6, 7) na ViewController e altere o atributo System Item para Save.



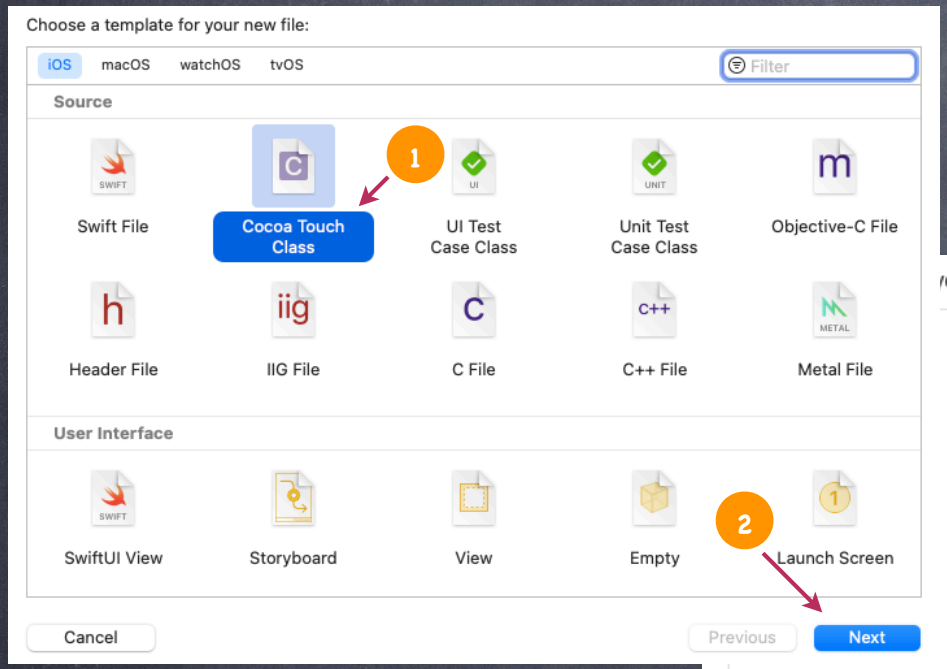
ViewController

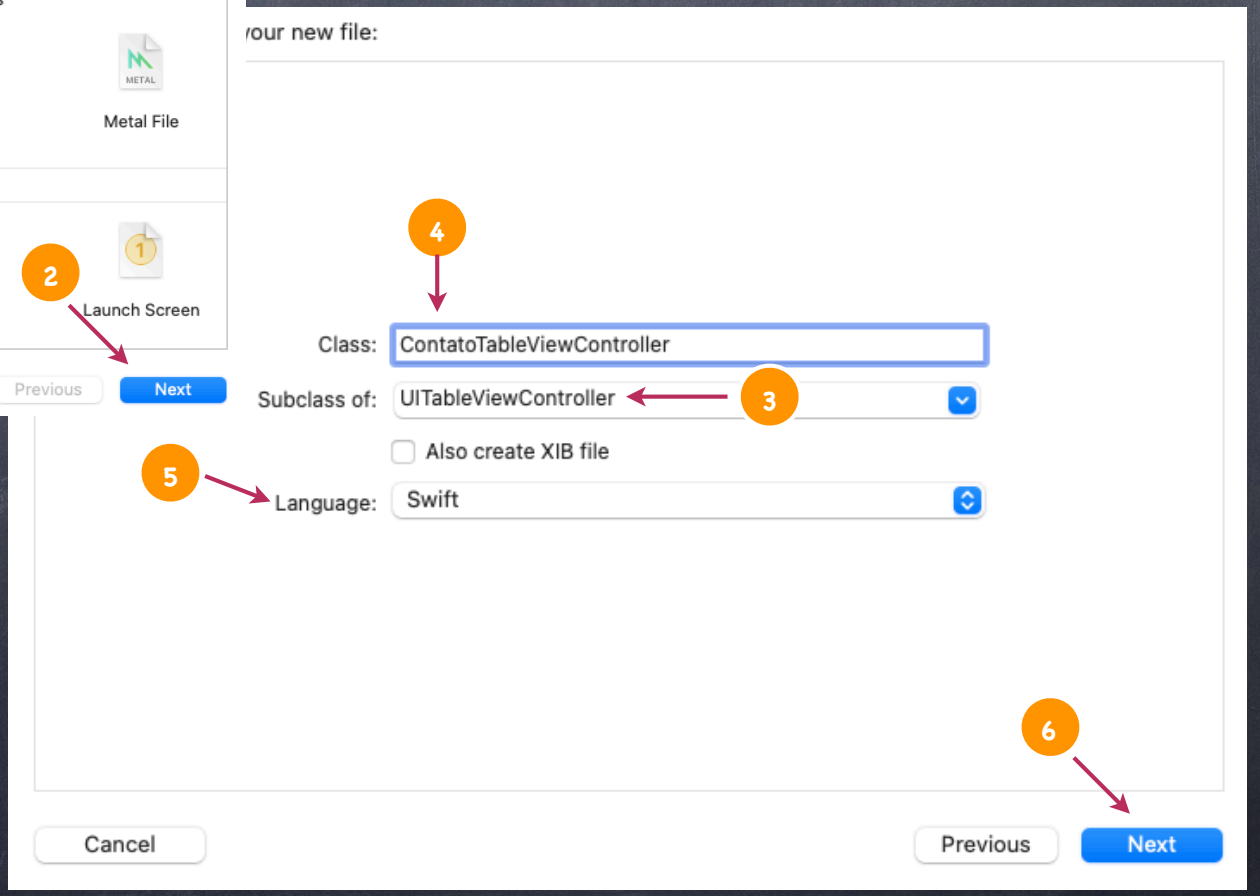
- Insira na ViewController três campos do tipo Text Field (1,2,3) e altere a propriedade Placeholder (4) de cada um.



Classe TableViewController

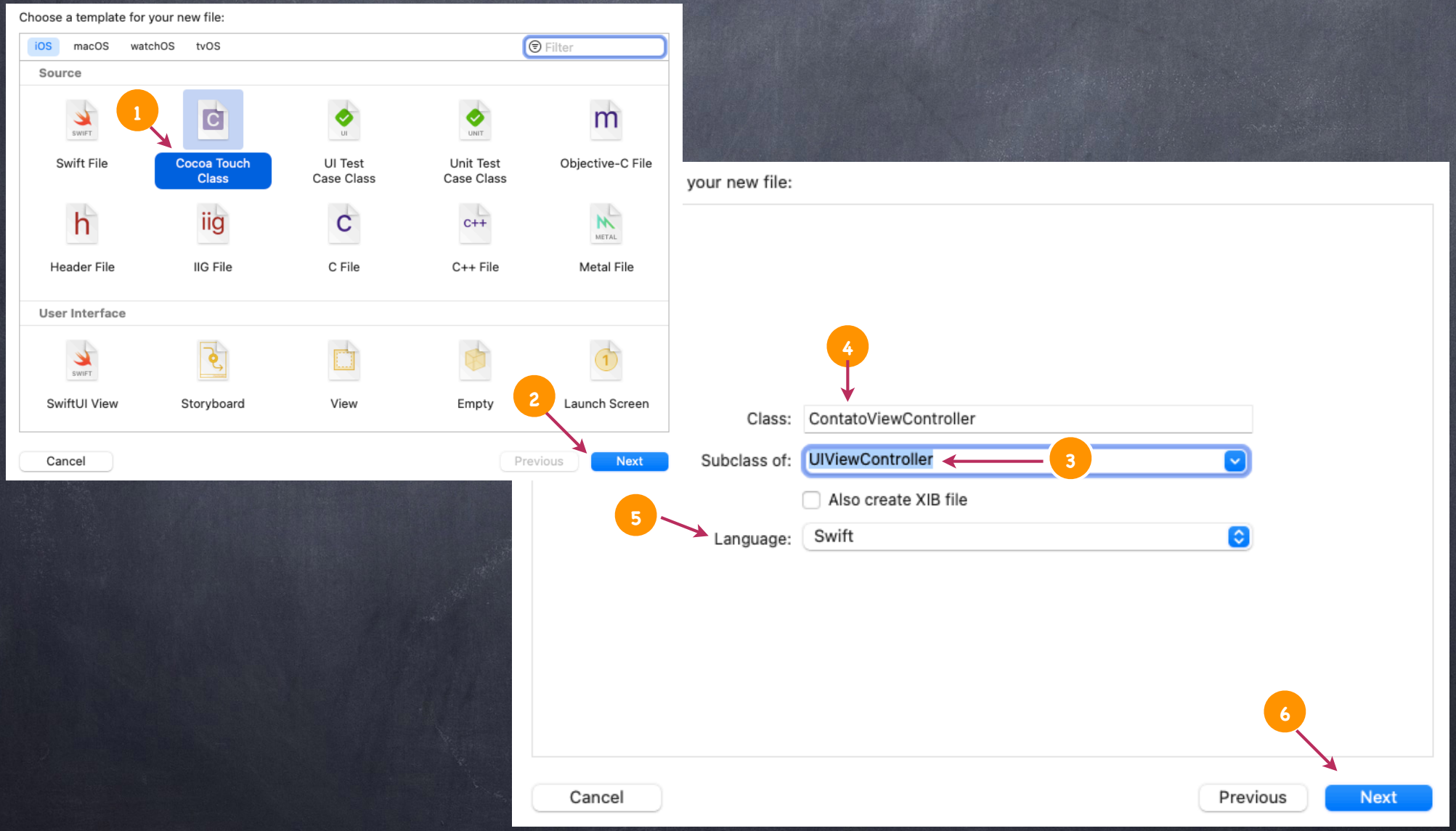
- Adicione uma nova Classe (Command + N), escolha subclasse de UITableViewController e nomeie como ContatoTableViewController.





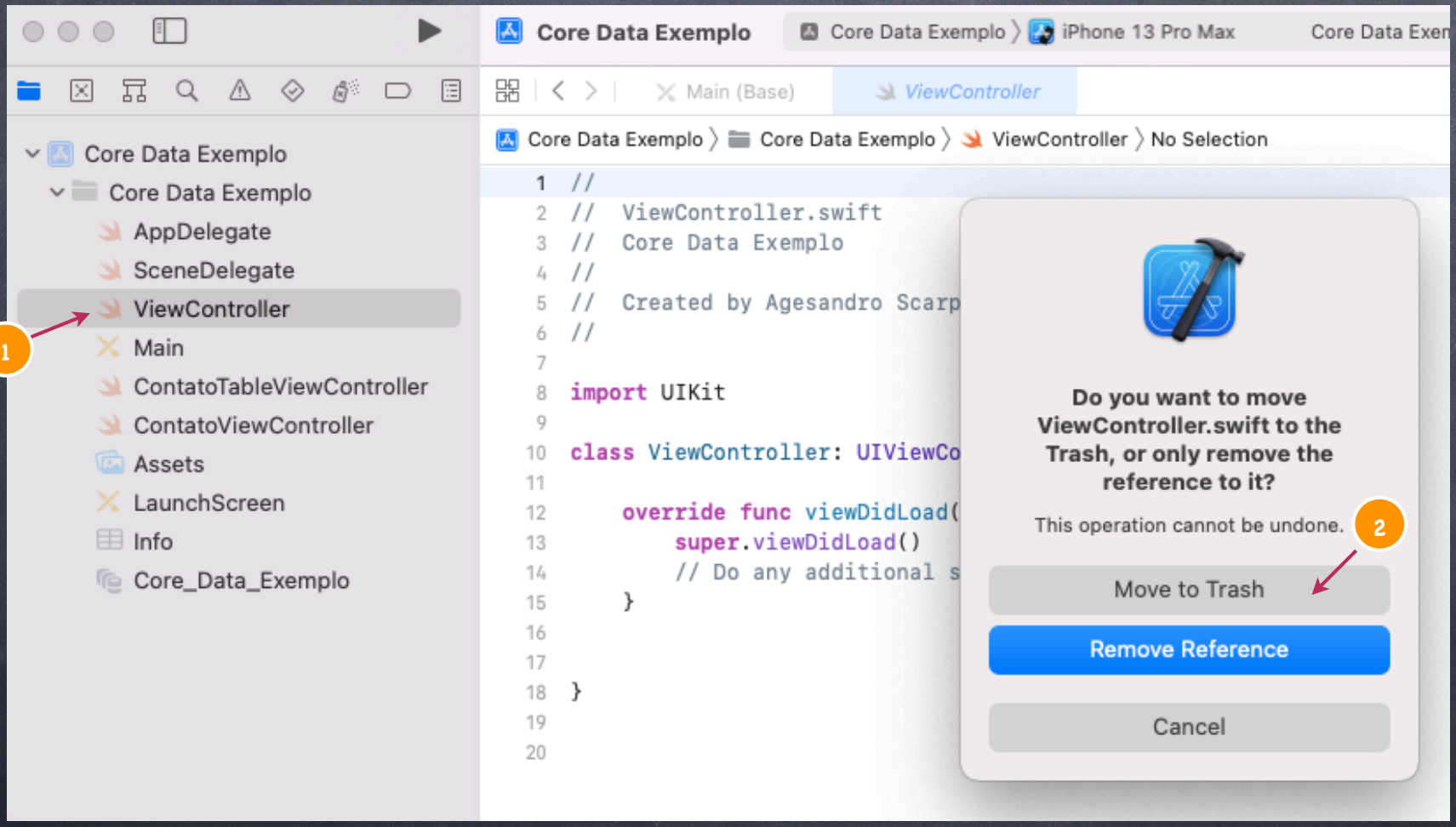
Classe ViewController

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



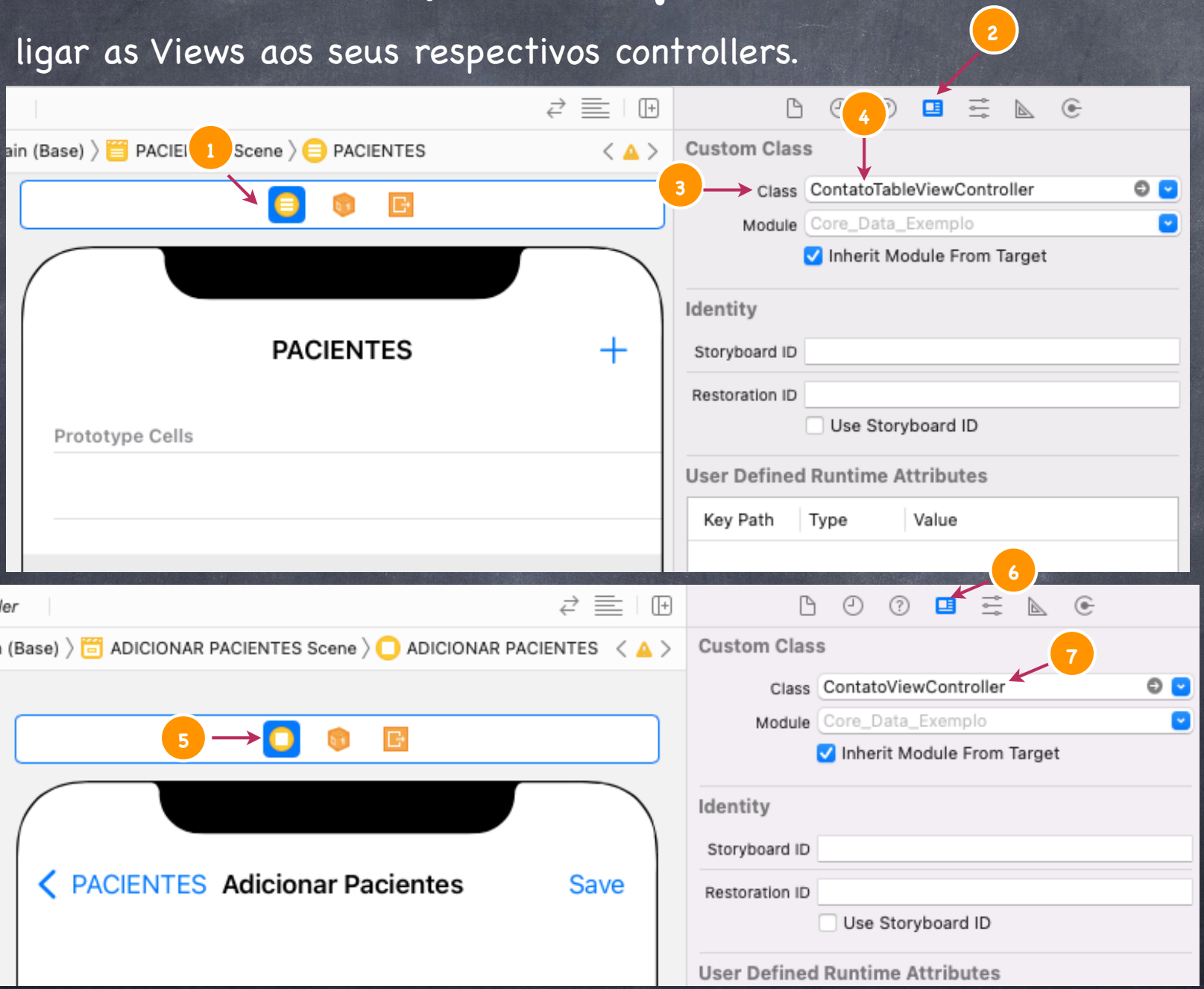
Apagando ViewController.swift

- Selecione a ViewController(1) que foi criada junto com o projeto, com o botão direito escolha delete e clique em Move To Trash (2).



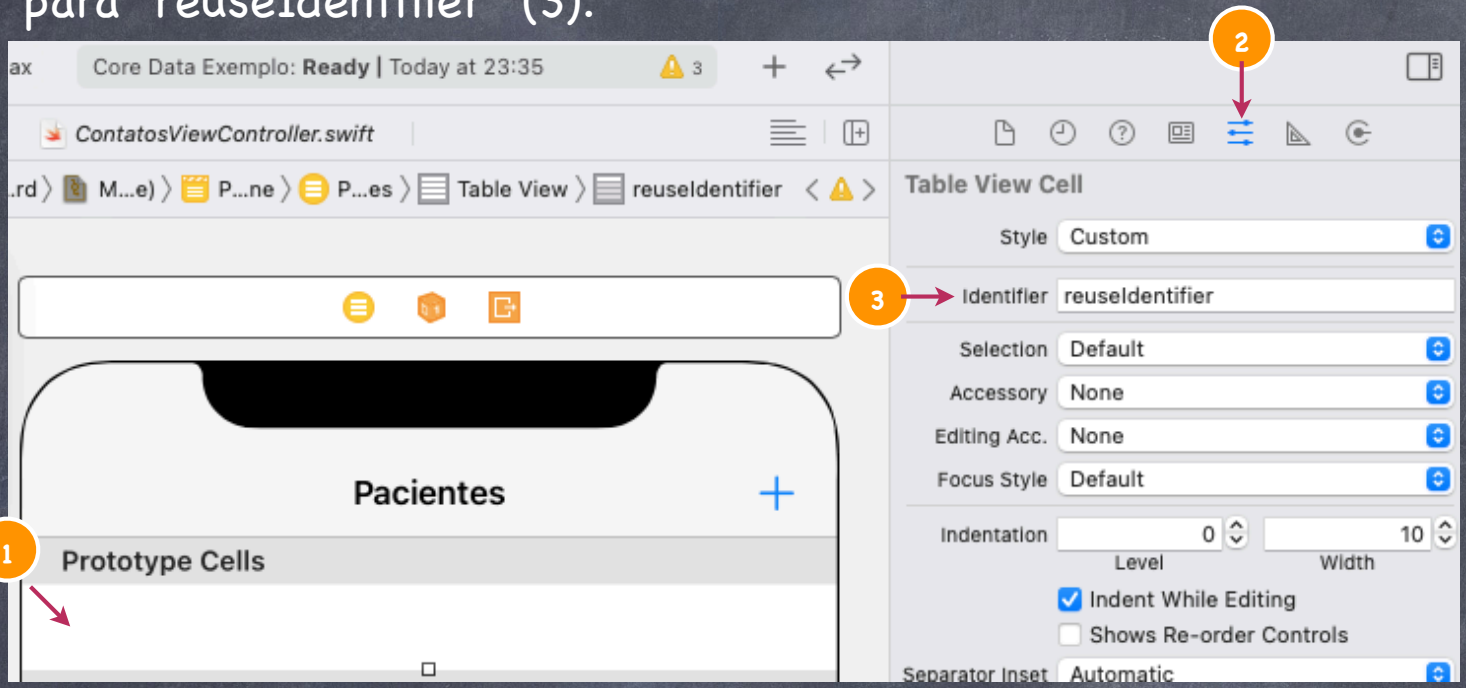
Identity Inspector

- Vamos ligar as Views aos seus respectivos controllers.

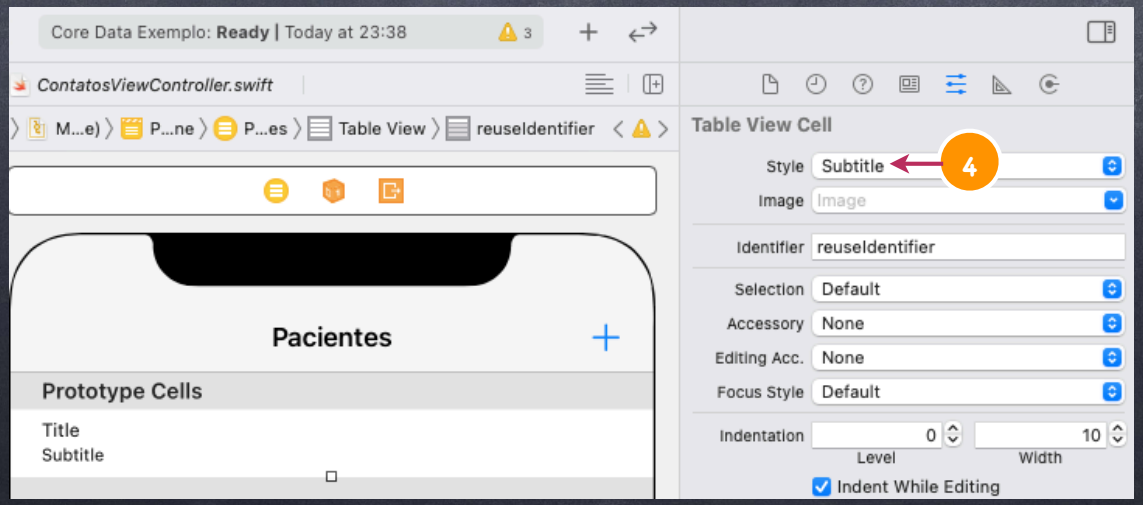


ReuseIdentifier

- Clica na Table View Cell (1), entre em Attributes Inspector (2) e altere o atributo Identifier para "reuseIdentifier" (3).

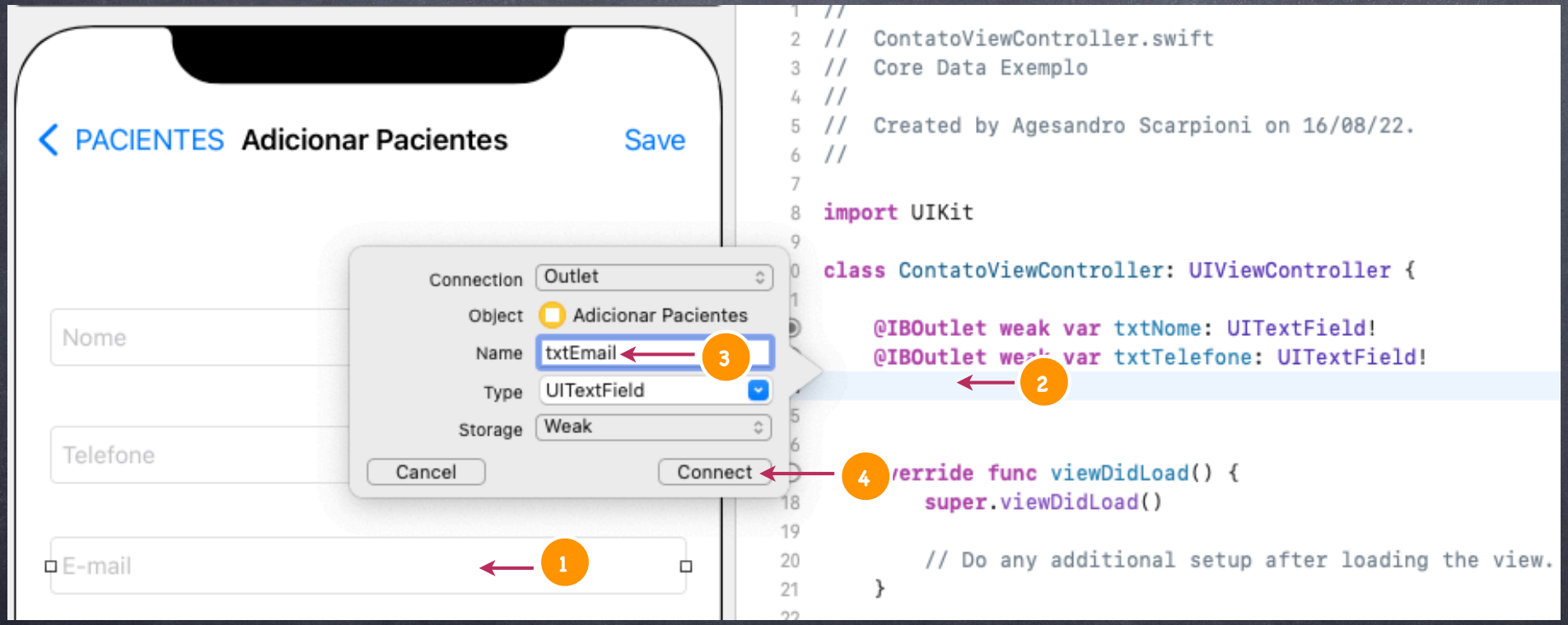


- Altere o atributo Style para Subtitle (4).



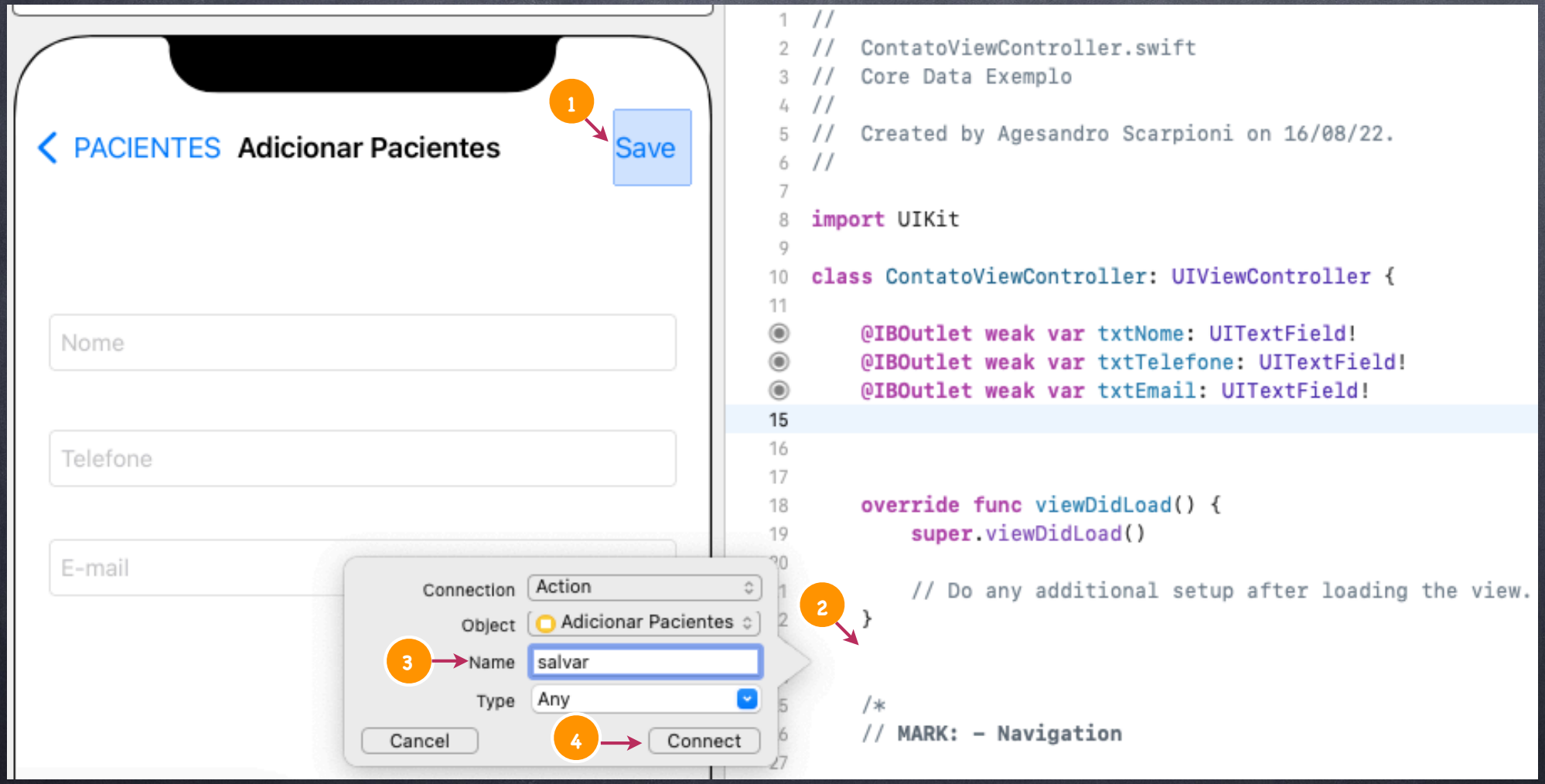
Text Field

- Vamos deixar abertos simultaneamente o Storyboard e o ContatoViewController, crie os três Outlet's das respectivas caixas.



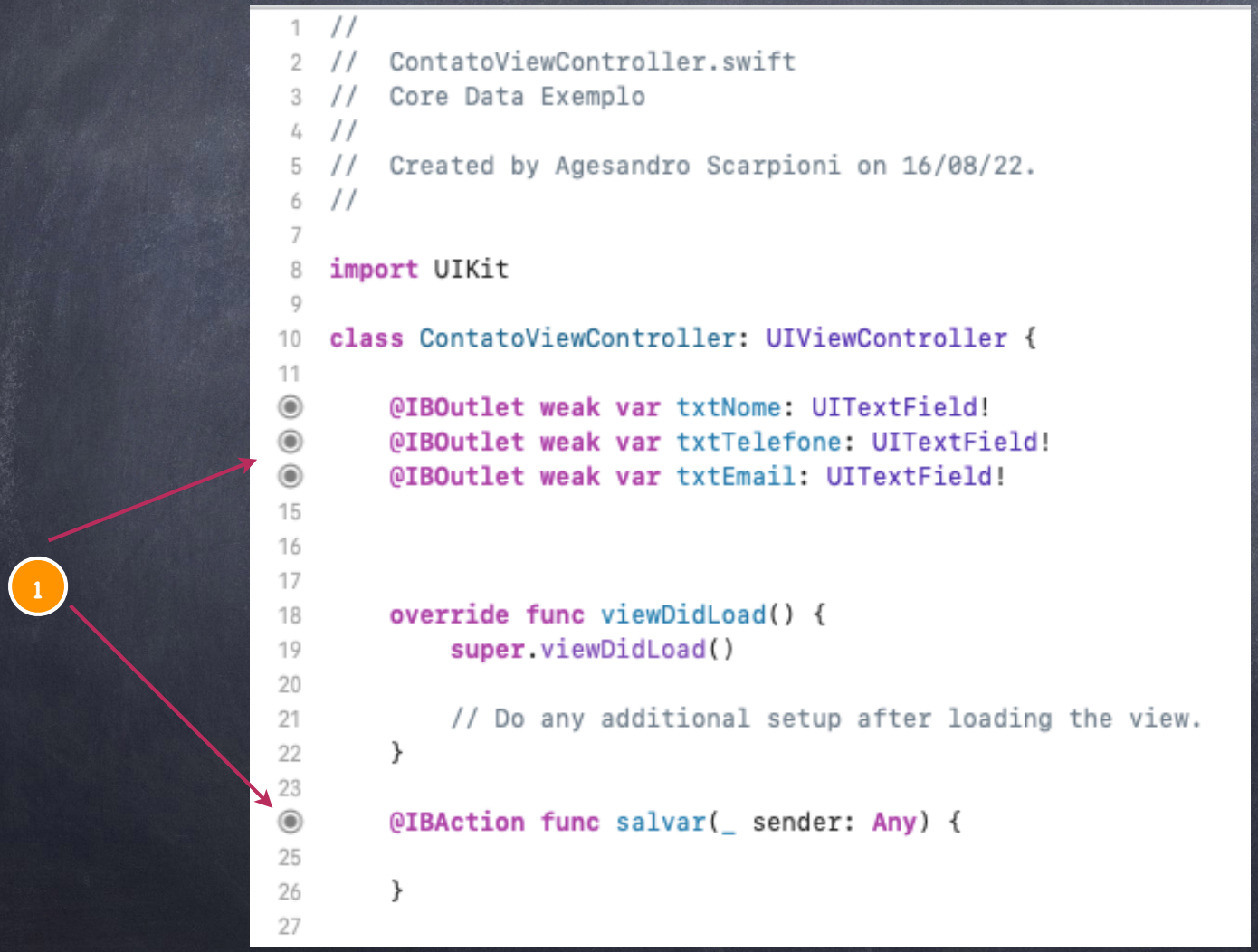
Action Salvar

- Ainda no ContatoViewController crie o IBAction para o botão Save.



Action e Outlet

- Veja seus Outlet's e o Action como no exemplo abaixo(1).



FIAP

Core Data

Criação da entidade e os atributos para armazenamento
Parte 2

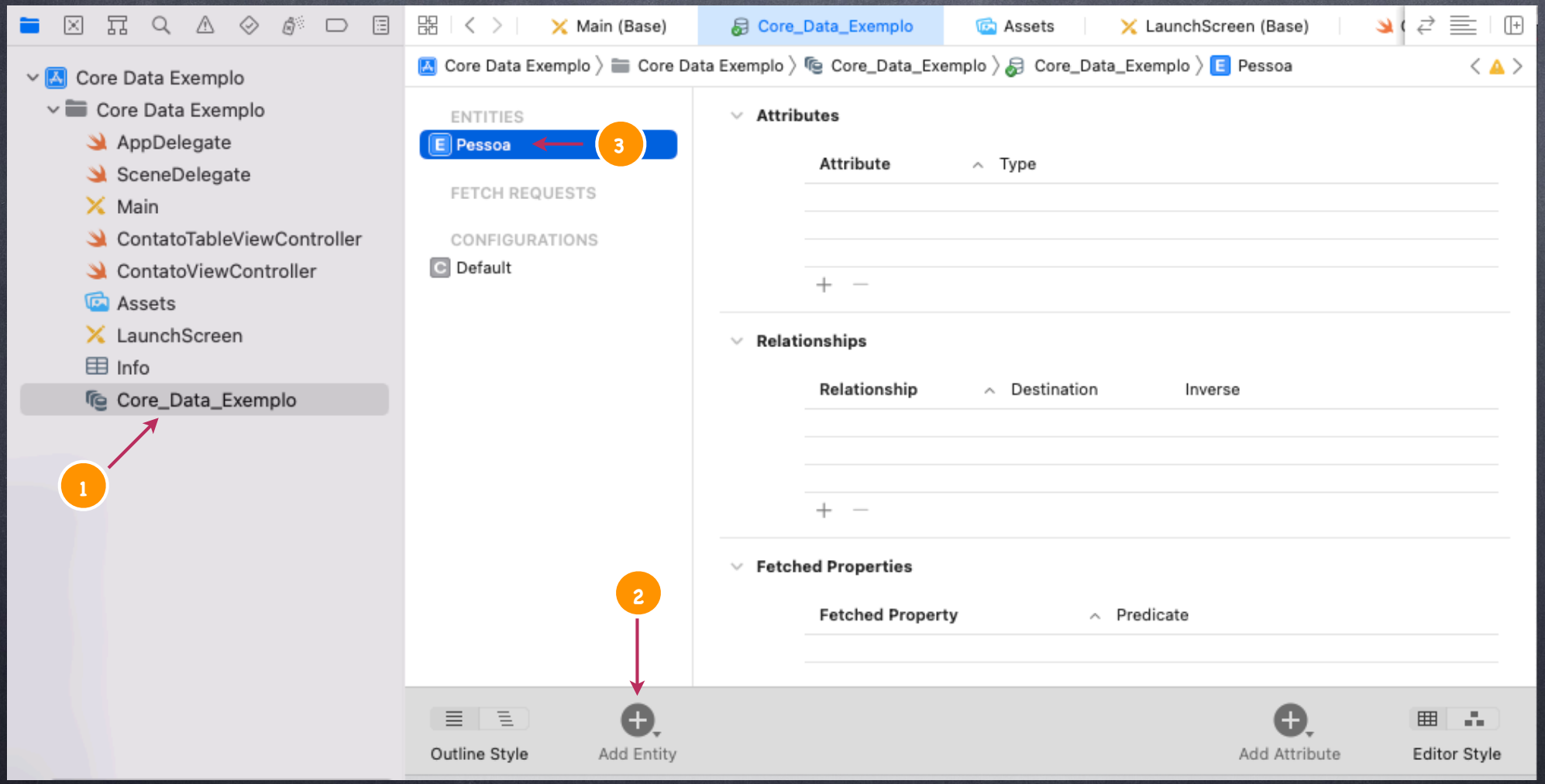
FIAP

Core Data

- A partir desse ponto você irá criar a entidade Pessoa. (Esta entidade irá aparecer no código como um NSManageObject)
- Irá criar os atributos: Nome, Telefone e E-mail todos como String (Atributos são acessados no NSManagedObject via métodos valueForKey: e setValueForkey:.
- O editor de modelo de dados tem muitos recursos que você pode explorar mais tarde. Por enquanto, vamos nos concentrar na criação de uma única entidade de dados centrais

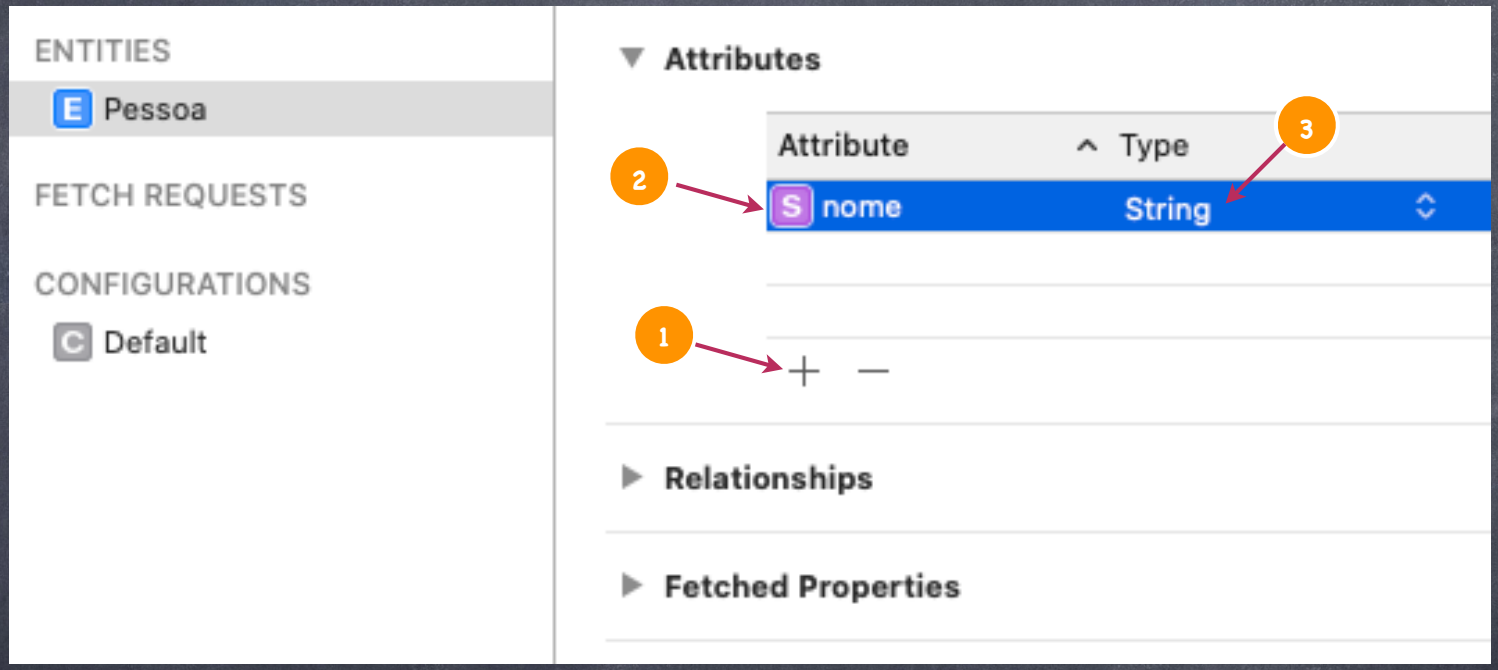
Core Data Model

- Abra o editor de modelo de dados Core_Data_Exemplo(1), clique em Add Entity(2) para criar uma nova entidade. Clique duas vezes na nova entidade e altere seu nome para Pessoa(3).



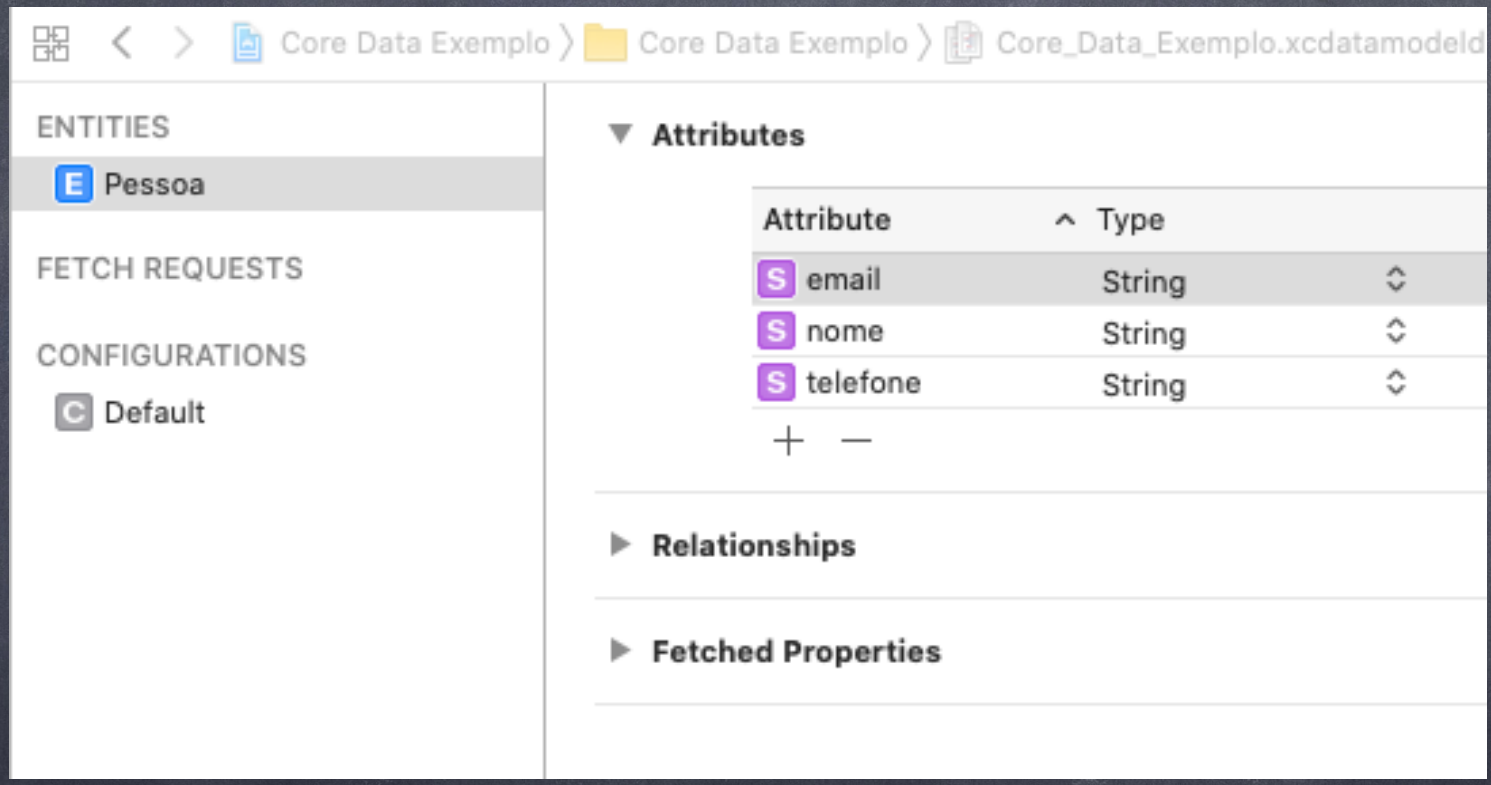
Core Data Model

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



Core Data Model

- Esse será o resultado após inserir a entidade Pessoa e todos seus atributos.



FIAP

Core Data

Salvando os dados com Core Data
Parte 3

Core Data

- Na classe ContatoViewControllerFaça o import do CoreData(1) para ter acesso ao framework de persistência de dados.

```
1  //
2  //  ContatoViewController.swift
3  //  Core Data Exemplo
4  //
5  //  Created by Agesandro Scarpioni on 16/08/22.
6  //
7
8  import UIKit
9  import CoreData
10
11 class ContatoViewController: UIViewController {
12
13     @IBOutlet weak var txtNome: UITextField!
14     @IBOutlet weak var txtTelefone: UITextField!
15     @IBOutlet weak var txtEmail: UITextField!
16
17     override func viewDidLoad() {
18         super.viewDidLoad()
19
20         // Do any additional setup after loading the view.
21     }
```

1

Função Save

- Crie uma função save que receba as três caixas de texto da ViewController.

```swift
func save(name: String, tele:String, emai:String) {
  //AppDelegate da aplicação, acessa as linhas de códigos em AppDelegate.swift
  //que ficam disponíveis após você marcar o checkbox "Use Core Data" no início do projeto
  //Como sabemos que o contêiner está configurado no appDelegate.swift, precisamos referenciar esse contêiner
  guard let appDelegate = UIApplication.shared.delegate as? AppDelegate else { return }

  // Antes de salvar ou recuperar dados no Core Data você precisa de um managedObjectContext,
  //considere-o como um "bloco de notas" na memória para trabalhar com objetos gerenciados.
  //Ele é usado para salvar, atualizar, deletar e buscar objetos
  //vamos criar um contexto (gerenciador de objetos de contexto) para esse container
  let managedContext = appDelegate.persistentContainer.viewContext

  // Aqui você cria uma nova instância de entidade pessoa (inserindo no contexto do objeto gerenciado).
  // em resumo uma entidade para novos registros
  let entidade = NSEntityDescription.entity(forEntityName: "Pessoa", in: managedContext)!

  //para cada insert é necessário um conjunto de linhas desses.
  //imagine se houvesse um loop gravando várias informações, esse
  //trecho abaixo //---inicio do ... até //---Fim do... estaria dentro do loop

  //---Início do processo de gravação
  let pessoa = NSManagedObject(entity: entidade, insertInto: managedContext)

  //Com um ManagedObject em mãos, você define o atributo de nome usando a codificação de valor-chave,
  //que devem ser idênticos aos campos criados no modelo de dados
  pessoa.setValue(name, forKeyPath: "nome")
  pessoa.setValue(tele, forKeyPath: "telefone")
  pessoa.setValue(emai, forKeyPath: "email")

  //Você confirma suas alterações, invocando o método "save" no
  //contexto do objeto gerenciado para comitar a mudança
  do {
    try managedContext.save()
  } catch let error as NSError {
    print("Não foi possível salvar. \(error), \(error.userInfo)")
  }
  //---Fim do processo de gravação
}
```

Action Salvar

- No Action salvar, chame a função save e passe as três caixas de texto, em seguida feche a janela que está aberta. Execute seu App e cadastre duas pessoas para um teste rápido.

```
24
25      @IBAction func salvar(_ sender: Any) {
26          //chamando o método save e passando as caixas de texto como parâmetro
27          self.save(name: txtNome.text!, tele: txtTelefone.text!, emai: txtEmail.text!)
28          //após salvar, a view será fechada e por consequência do didAppear na outra tela, fazemos um reload no Table
29          self.navigationController?.popViewController(animated: true)
30      }
31
```

FIAP

Core Data

Programando o TableViewController para exibir os dados

Parte 4

Core Data
Consulta

FIAP

- O processo de obtenção dos dados salvos é muito simples.
- Prepare a solicitação do tipo NSFetchRequest para a entidade Pessoa em nosso exemplo.
- Use fetchLimit para definir a quantidade de registros que será buscada
- Use o Predicate caso necessite filtrar os dados
- Use SortDescriptors para ordenar os dados.
- Obtenha o resultado do contexto na forma de array de [NSManagedObject]
- Pode ser feito uma iteração em uma matriz para obter o valor da chave específica

Consulta

- Em ContatosTableViewController, inclua uma linha de import do framework CoreData(1) e um getset do tipo NSManagedObject para pessoas(2).

- Um NSManagedObject(2) pode assumir qualquer entidade no modelo de dados apropriando-se de qualquer atributo e relacionamento definido no modelo, você irá utilizá-lo para criar, editar, excluir registros no Core Data.

```
1  //
2  //  ContatosTableViewController.swift
3  //  Core Data Exemplo
4  //
5  //  Created by Agesandro Scarpioni on 25/08/20.
6  //  Copyright © 2020 Agesandro Scarpioni. All rights reserved.
7  //
8
9  import UIKit
10 import CoreData
11
12 class ContatosTableViewController: UITableViewController {
13
14     var pessoas: [NSManagedObject] = []
15
```

1

2

Consulta

- Inclua a função viewWillAppear abaixo, para buscar os dados do Core Data.

```swift
override func viewWillAppear(_ animated: Bool) {
    //Como sabemos que o container está configurado no appDelegate, precisamos referenciar esse container
    guard let appDelegate = UIApplication.shared.delegate as? AppDelegate else {return}
    //Vamos criar um contexto (Gerenciador de objetos de contexto) para esse container
    let managedContext = appDelegate.persistentContainer.viewContext

    //Como o nome sugere, NSFetchRequest é a classe responsável pela busca dos dados.
    //Ela irá preparar a requisição para a entidade Pessoa.
    let fetchRequest = NSFetchRequest<NSManagedObject>(entityName: "Pessoa")

    //Quantidade que deve ser exibida
    //fetchRequest.fetchLimit = 1

    //Qual campo e dado que deve ser procurado
    //fetchRequest.predicate = NSPredicate(format: "nome = %@", "teste")

    //Qual campo e ordem (crescente ou decrescente) que deve ser exibida a informação
    fetchRequest.sortDescriptors = [NSSortDescriptor.init(key: "nome", ascending: true)]

    //Você entrega a solicitação de busca ao contexto do objeto gerenciado, que no caso
    //é nosso "pessoas", ou seja, na forma de um array de NSManagedObject
    do {
        pessoas = try managedContext.fetch(fetchRequest)
    } catch let error as NSError {
        print("Não foi possível buscar os dados \(error), \(error.userInfo)")
    }
    //Toda vez que aparece o TableView um reload da Table é executado para
    //exibir o último registro digitado, ou qualquer alteração feita na lista
    self.tableView.reloadData()
}
```

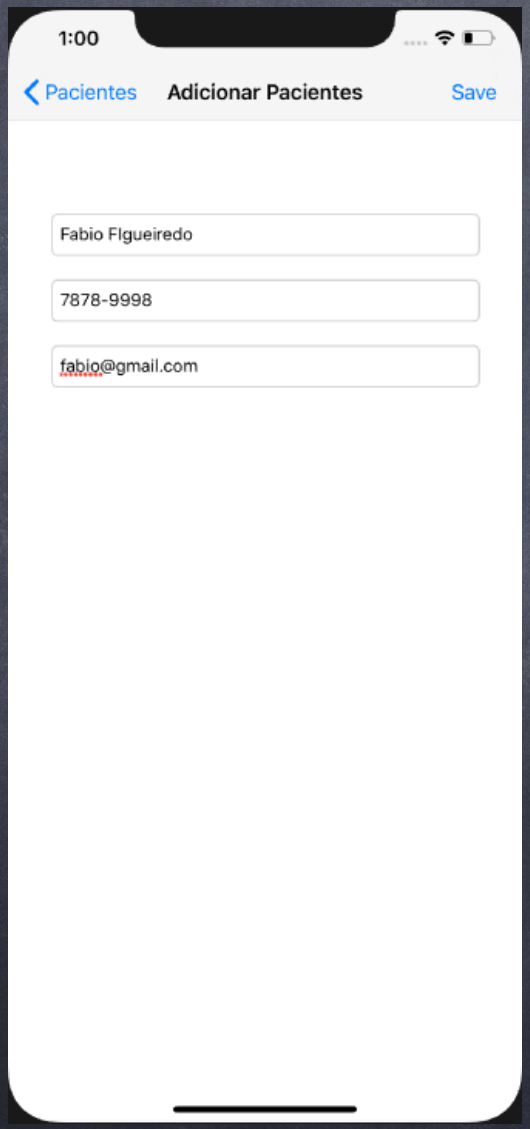
Populando a TableView

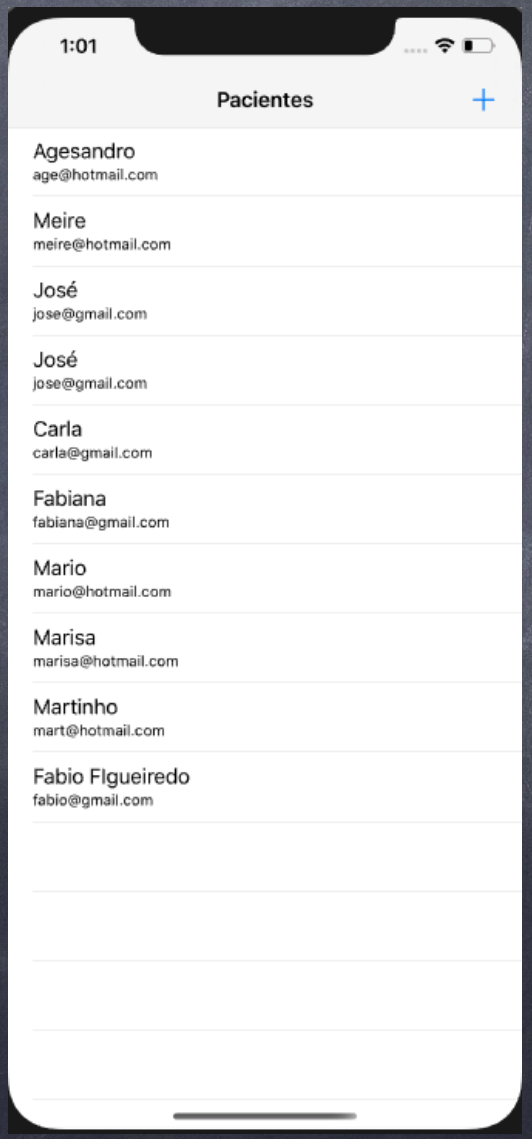
- Programe os três métodos abaixo para exibir os dados que foram buscados no método viewWillAppear

```
65
66      // MARK: - Table view data source
67
68      override func numberOfSections(in tableView: UITableView) -> Int {
69          // #warning Incomplete implementation, return the number of sections
70          return 1
71      }
72
73      override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
74          // #warning Incomplete implementation, return the number of rows
75          return pessoas.count
76      }
77
78      override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
79          let cell = tableView.dequeueReusableCell(withIdentifier: "reuseIdentifier", for: indexPath)
80          let pessoa = pessoas[indexPath.row]
81          // Configure the cell...
82          cell.textLabel?.text = pessoa.value(forKeyPath: "nome") as? String
83          cell.detailTextLabel?.text = pessoa.value(forKey: "email") as? String
84          return cell
85      }
```

Testando

- Execute seu projeto (Command + R) e salve alguns nomes, ainda precisamos melhorar a interação com as caixas de texto, mas os dados já podem ser salvos e exibidos imediatamente na TableView.





FIAP

Core Data

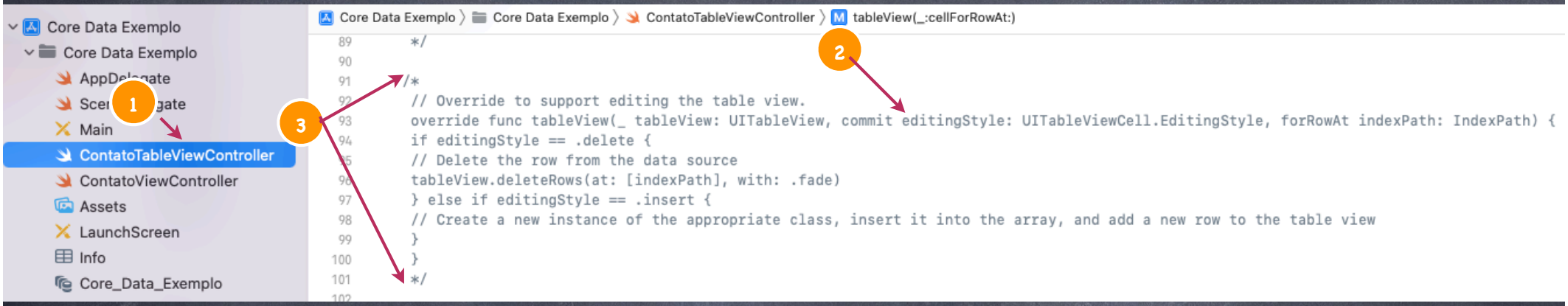
Programando o TableViewController para apagar dados
Parte 5

Core Data
Exclusão

- Para excluir um registro, primeiro temos que encontrar o objeto que queremos excluir por fetchRequest
- Preparar a solicitação com predicado para a entidade Pessoa que é o exemplo deste slide
- Obter o registro que iremos apagar
- Fazer a chamada context.delete (objeto)
- Como o objeto que queremos apagar nesta aula está na TableView utilizaremos sua posição para excluí-lo, sendo assim não precisamos fazer a busca por fetchRequest, ou seja, o registro já está carregado em NSManaged

Método Commit EditingStyle

- Em ContatoTableViewController(1), localize o método commit EditingStyle (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.



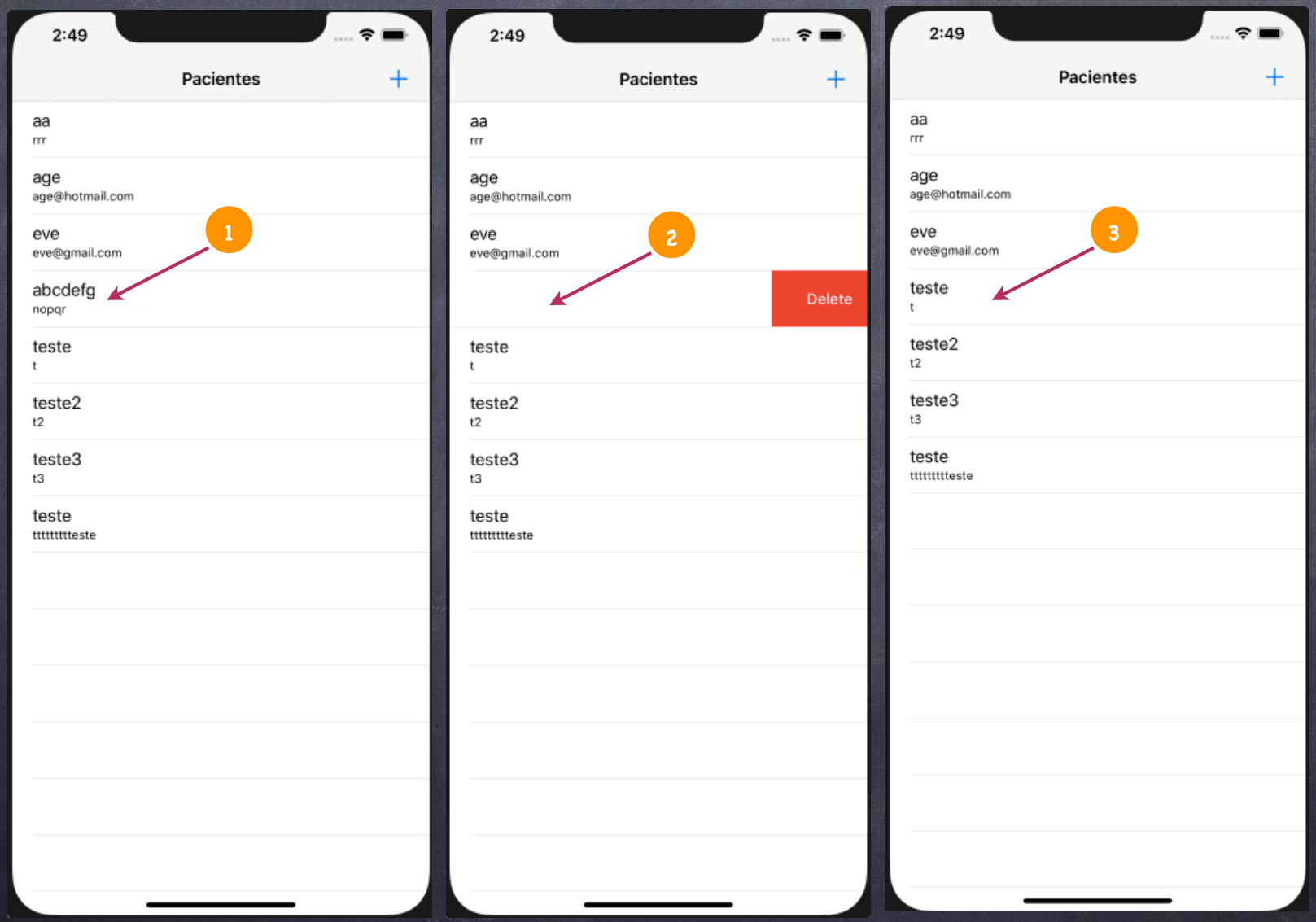
Método Commit EditingStyle

- Para excluir um registro, digite as linhas 103 até a 114, exceto a linha 111 e todo o restante que já pertencia no código que estava comentado.

```
97
98      // Override to support editing the table view.
99      override func tableView(_ tableView: UITableView, commit editingStyle: UITableViewCell.EditingStyle, forRowAt indexPath: IndexPath) {
100         if editingStyle == .delete {
101             // Delete the row from the data source
102             //
103             guard let appDelegate = UIApplication.shared.delegate as? AppDelegate else { return }
104             let managedContext = appDelegate.persistentContainer.viewContext
105
106             managedContext.delete(pessoas[indexPath.row])//deletando o ítem de pessoas do Contexto
107
108             do {
109                 try managedContext.save()//fazendo o Commit dessa exclusão
110                 pessoas.remove(at: indexPath.row)//retirando o item de pessoas nosso NSManagedObject
111                 tableView.deleteRows(at: [indexPath], with: .fade) //Apagando a linha do Table, essa linha já estava no código
112             } catch let error as NSError {
113                 print("Não foi possível excluir. \(error), \(error.userInfo)")
114             }
115         } else if editingStyle == .insert {
116             // Create a new instance of the appropriate class, insert it into the array, and add a new row to the table view
117         }
118     }
119
```

Testando

- Execute seu projeto (Command + R) e apague algumas linhas



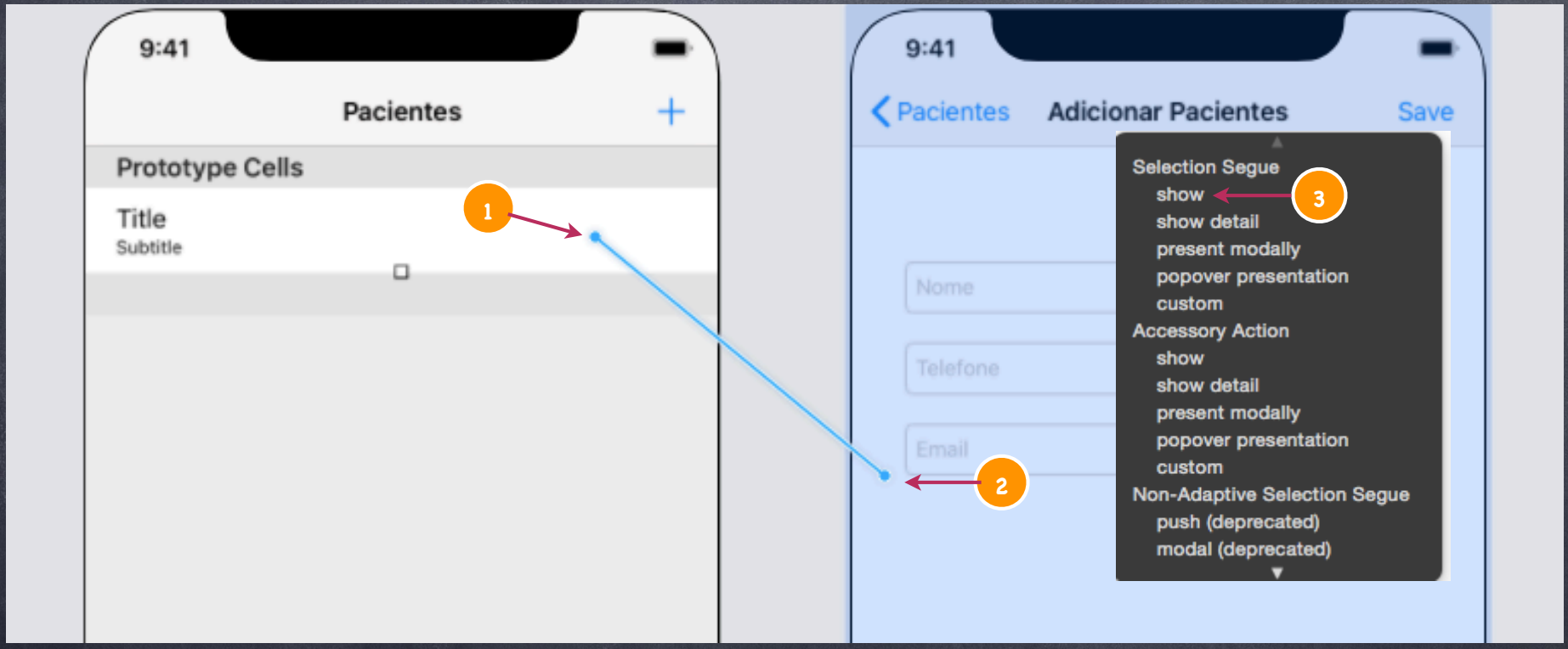
Core Data
Edição

FIAP

- Para atualizar um registro, primeiro você precisa buscar o registro que deve ser editado.
- Prepare a solicitação com predicado para a entidade Pessoa que é o exemplo deste slide.
- Obter registro e definir novo valor com a chave
- Por último salvar o contexto
- Como o objeto que queremos editar nesta aula está na TableView utilizaremos sua posição para abrir a janela de cadastro, movimentar os dados para ela e de lá fazer a edição, sendo assim não precisamos fazer a busca por fetchRequest, ou seja, o registro já está carregado em NSManaged

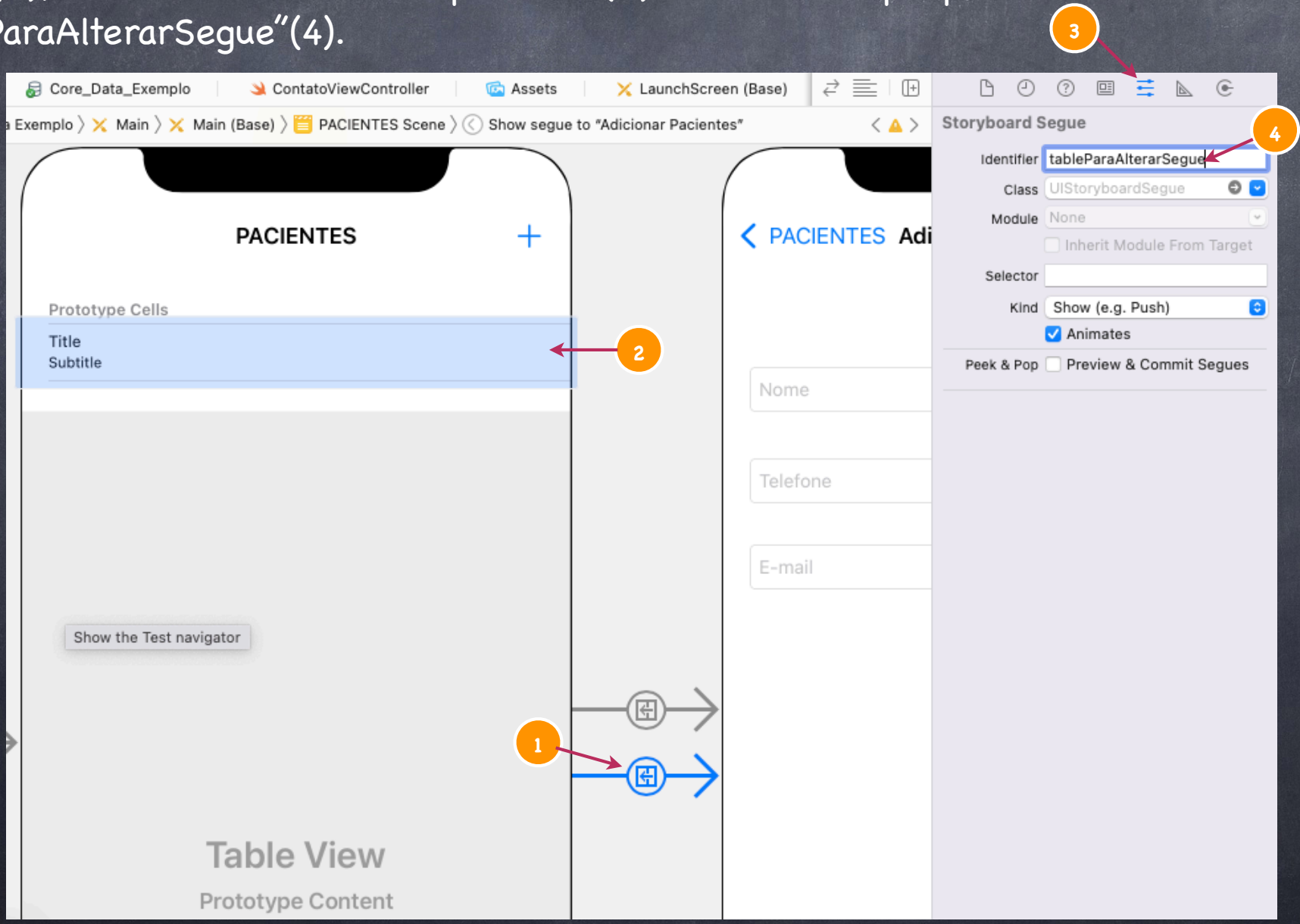
Core Data Edição Tela

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



Identificar a Segue

- Selecione a Segue que você acabou de criar (1), observe se a Segue é referente a célula(2), em Attributes Inspector (3) altere a propriedade Identifier para: "tableParaAlterarSegue"(4).



NSManagedObject

- Em ContatoViewController(1) adicione um NSManagedObject(2), para receber os dados vindos do click na célula da TableView.

```swift
//
//  ContatoViewController.swift
//  Core Data Exemplo
//
//  Created by Agesandro Scarpioni on 16/08/22.
//

import UIKit
import CoreData

class ContatoViewController: UIViewController {

    @IBOutlet weak var txtNome: UITextField!
    @IBOutlet weak var txtTelefone: UITextField!
    @IBOutlet weak var txtEmail: UITextField!

    var pessoaVindoDaTable:NSManagedObject?=nil

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
     }
```

1

2

Exibindo os dados

- Em ContatoViewController, faça a programação do IF para exibir os dados da passagem da Table para essa View quando houver conteúdo em "pessoaVindoDaTable".

```
20      override func viewDidLoad() {
21          super.viewDidLoad()
22          // Do any additional setup after loading the view.
23          if pessoaVindoDaTable != nil{
24              txtNome.text = pessoaVindoDaTable?.value(forKey: "nome") as? String
25              txtTelefone.text = pessoaVindoDaTable?.value(forKey: "telefone") as? String
26              txtEmail.text = pessoaVindoDaTable?.value(forKey: "email") as? String
27          }
28      }
```

If para Update

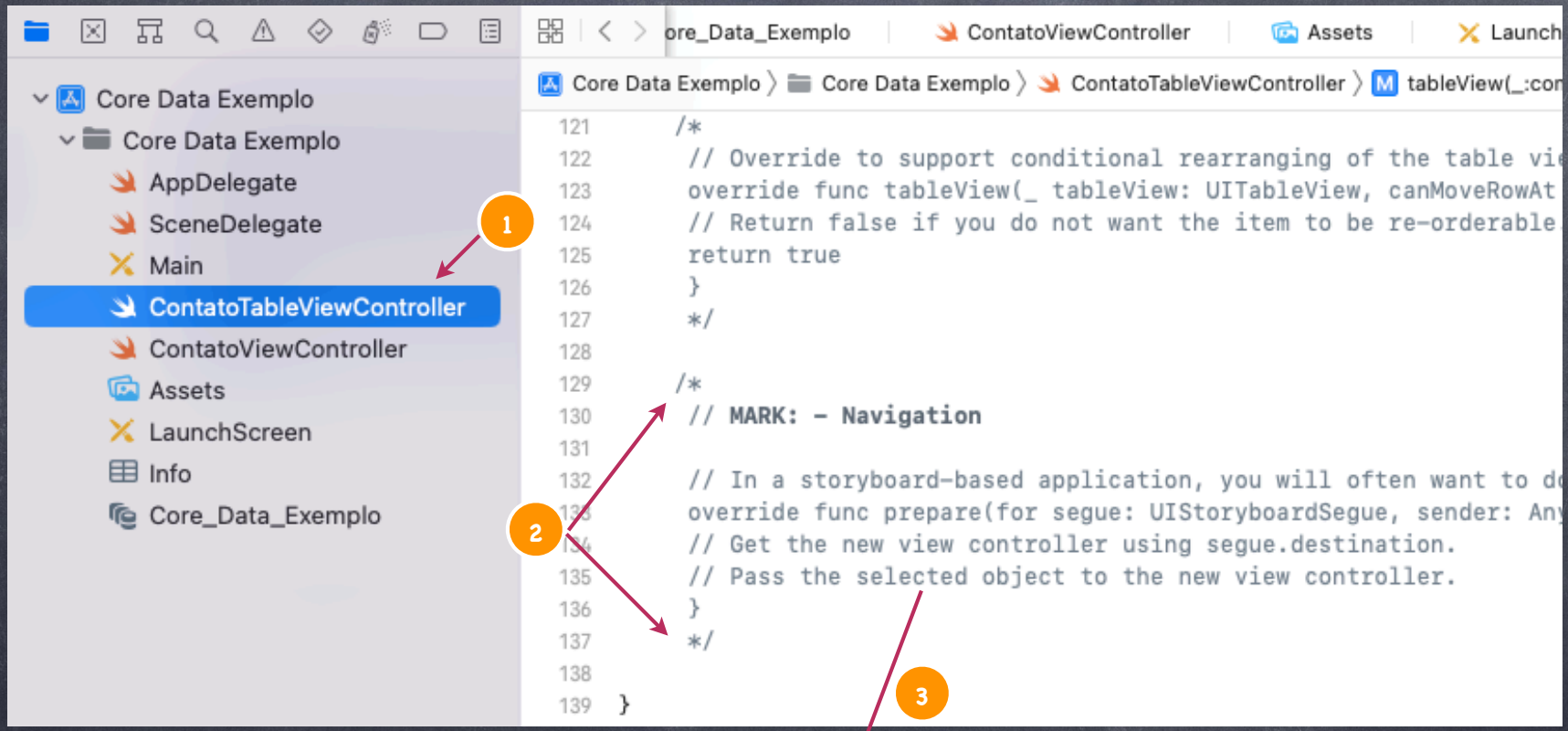
- Em ContatoViewController adicione o "if else" na linha 60 e na 67 ajustando a parte de insert que já existia, depois do else digite o código para Update (1), esse if servirá para diferenciar o update de um insert.

```
44      func save(name: String, tele:String, emai:String) {
45          //AppDelegate da aplicação, acessa as linhas de códigos em AppDelegate.swift
46          //que ficam disponíveis após você marcar o checkbox "Use Core Data" no início do projeto
47          //Como sabemos que o contêiner está configurado no appDelegate.swift, precisamos referenciar esse contêiner
48          guard let appDelegate = UIApplication.shared.delegate as? AppDelegate else { return }
49
50          // Antes de salvar ou recuperar dados no Core Data você precisa de um managedObjectContext,
51          //considere-o como um "bloco de notas" na memória para trabalhar com objetos gerenciados.
52          //Ele é usado para salvar, atualizar, deletar e buscar objetos
53          //vamos criar um contexto (gerenciador de objetos de contexto) para esse container
54          let managedContext = appDelegate.persistentContainer.viewContext
55
56          // Aqui você cria uma nova instância de entidade pessoa (inserindo no contexto do objeto gerenciado).
57          // em resumo uma entidade para novos registros
58          let entidade = NSEntityDescription.entity(forEntityName: "Pessoa", in: managedContext)!
59
60          if (pessoaVindoDaTable == nil){
61              let pessoa = NSManagedObject(entity: entidade, insertInto: managedContext)
62              //Com um ManagedObject em mãos, você define o atributo de nome usando a codificação de valor-chave,
63              //que devem ser idênticos aos campos criados no modelo de dados
64              pessoa.setValue(name, forKeyPath: "nome")
65              pessoa.setValue(tele, forKeyPath: "telefone")
66              pessoa.setValue(emai, forKeyPath: "email")
67          }else{
68              let objectUpdate = pessoaVindoDaTable
69              objectUpdate!.setValue(name, forKeyPath: "nome")
70              objectUpdate!.setValue(tele, forKeyPath: "telefone")
71              objectUpdate!.setValue(emai, forKeyPath: "email")
72          }
73          //Você confirma suas alterações, invocando o método "save" no
74          //contexto do objeto gerenciado para comitar a mudança
75          do {
76              try managedContext.save()
77          } catch let error as NSError {
78              print("Não foi possível salvar. \(error), \(error.userInfo)")
79          }
80      }
```

1

Passando os dados

- Vá em ContatoTableViewController(1) e retire as linhas de comentário do método prepare for Segue(2). Veja como irá ficar o método sem os comentários(3).




```
// MARK: - Navigation

// In a storyboard-based application, you will often want to do a little preparation before navigation
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
// Get the new view controller using segue.destination.
// Pass the selected object to the new view controller.
}
```

Passando os dados

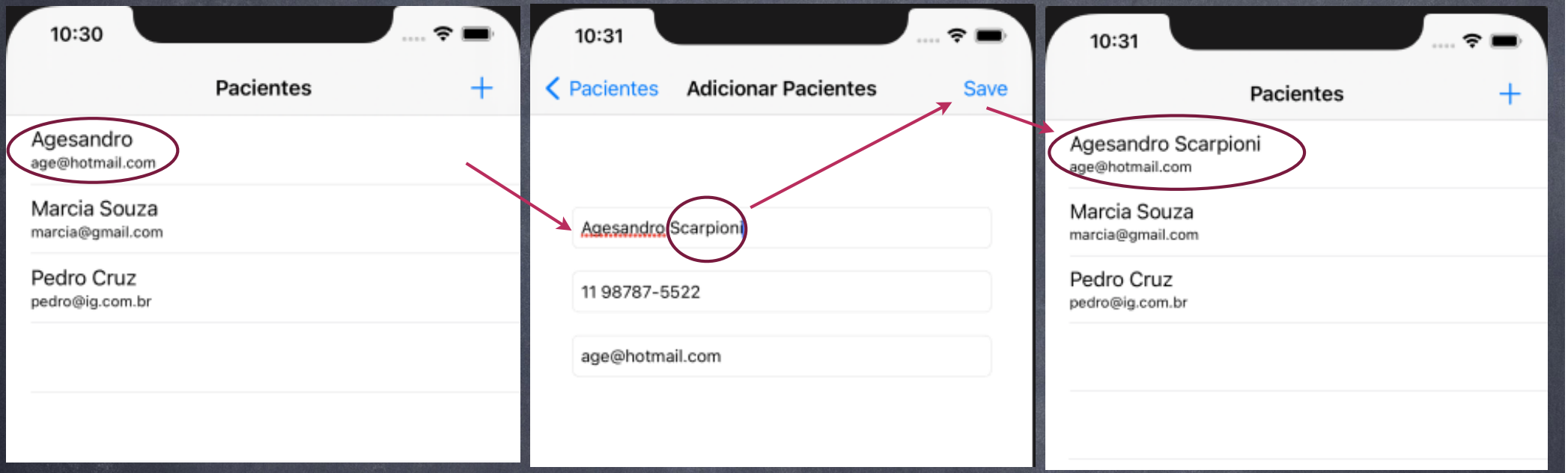
- Ainda em ContatoTableViewController programe a passagem dos dados do contatoTableViewController para o ContatoViewController

```
137     // MARK: - Navigation
138
139     // In a storyboard-based application, you will often want to do a little preparation before navigation
140     override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
141     // Get the new view controller using segue.destination.
142     // Pass the selected object to the new view controller.
143         if segue.identifier == "tableParaAlterarSegue"{
144             let vc = segue.destination as! ContatosViewController
145             let pessoaSelecionada:NSManagedObject = pessoas[self.tableView.indexPathForSelectedRow!.item]
146             vc.pessoaVindoDaTable = pessoaSelecionada
147         }
148     }
```

Obs: O nome tableParaAlterarSegue tem que ser exatamente igual ao nome dado no identifier da segue que criamos entre as duas views na pág. 42.

Core Data

- Agora vamos testar a edição



Prática

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