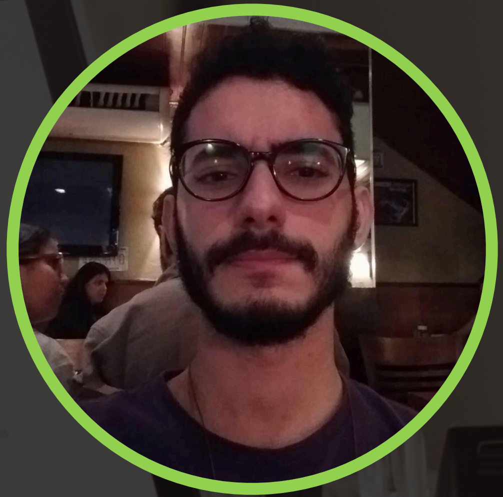
cit

Hilton **Pintor**

Desenvolvedor (iOS/tvOS/watchOS)



hiltonpintor@gmail.com

// Exercício 14

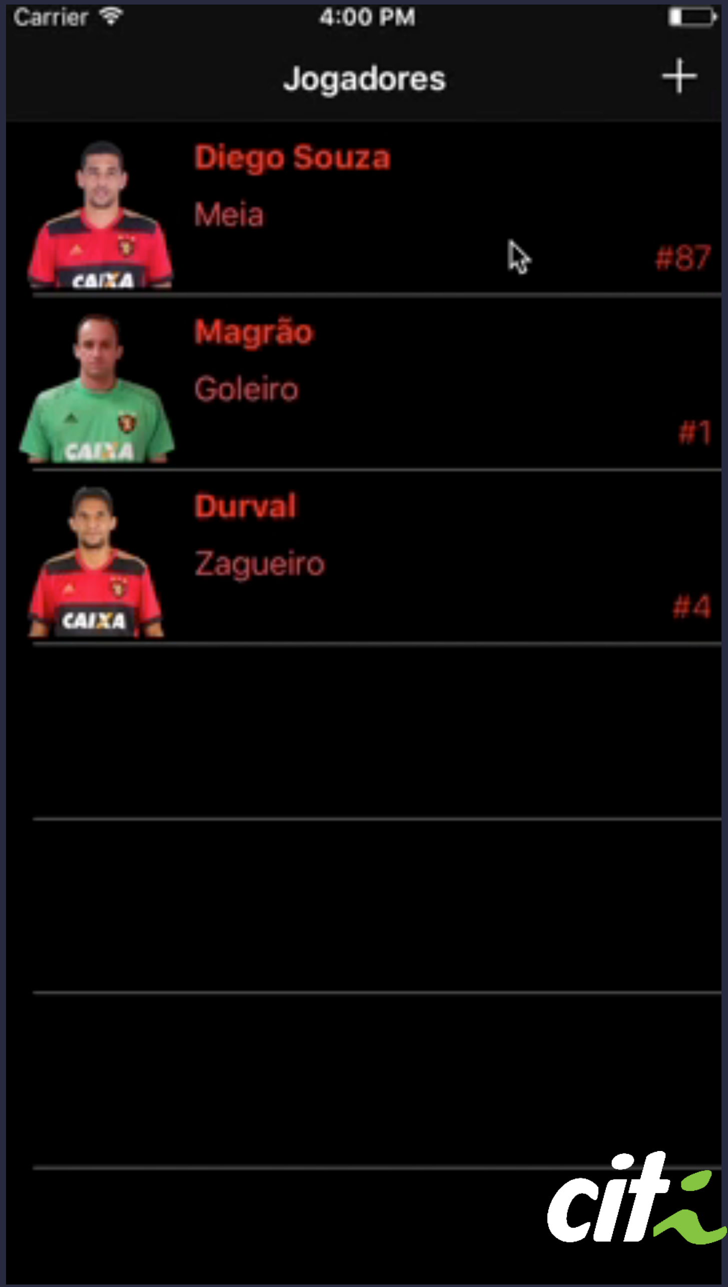
Lista de coisas II

1. ponha um botão de **adicionar**, que leva o usuário para uma tela onde ele escreverá um **nome**, e uma **foto** para um novo item a ser **inserido** na tabela tabela.

2. permita que o usuário possa **deletar** um item da tabela

// Extra

3. permita que o usuário possa **editar** um item da tabela



// Exercício 14

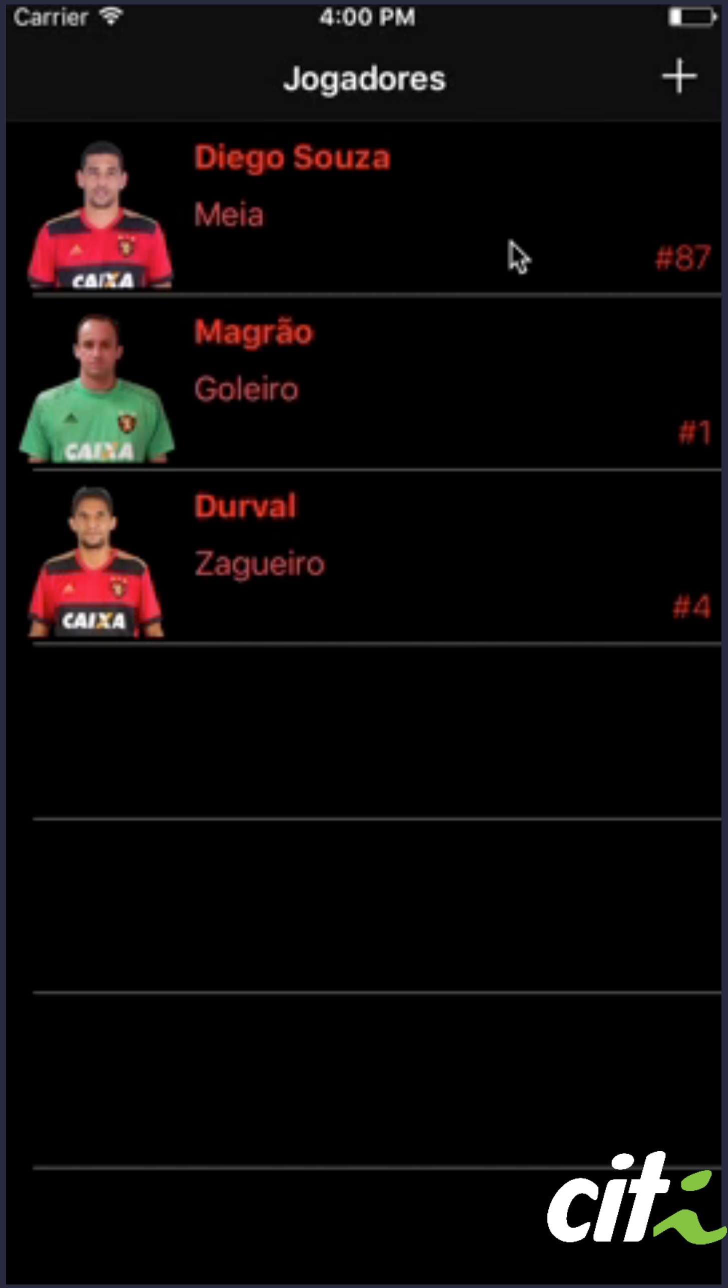
Lista de coisas II

1. ponha um botão de **adicionar**, que leva o usuário para uma tela onde ele escreverá um **nome**, e uma **foto** para um novo item a ser **inserido** na tabela tabela.

2. permita que o usuário possa **deletar** um item da tabela

// Extra

3. permita que o usuário possa **editar** um item da tabela



Diego Souza
Meia
#87

Magrão
Goleiro
#1

Durval
Zagueiro
#4

// Aula 08

// Dúvidas da Aula 07

/*

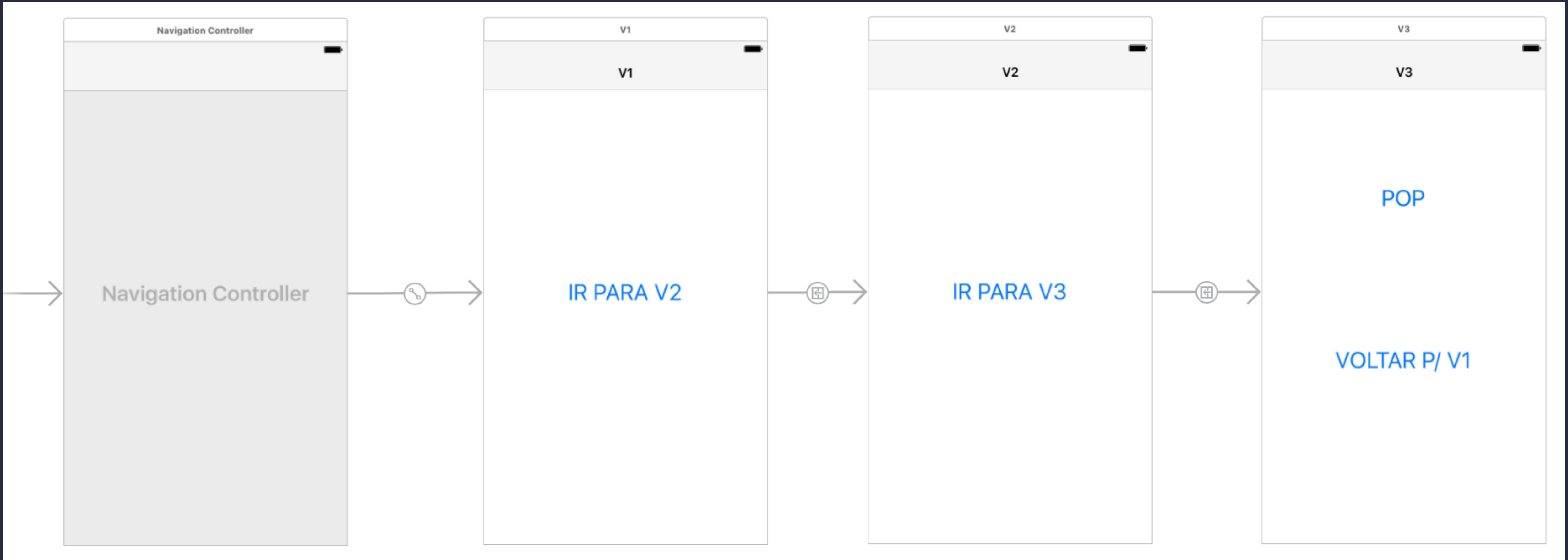
Como fazer unwind pelo código?

*/

// algoritmo

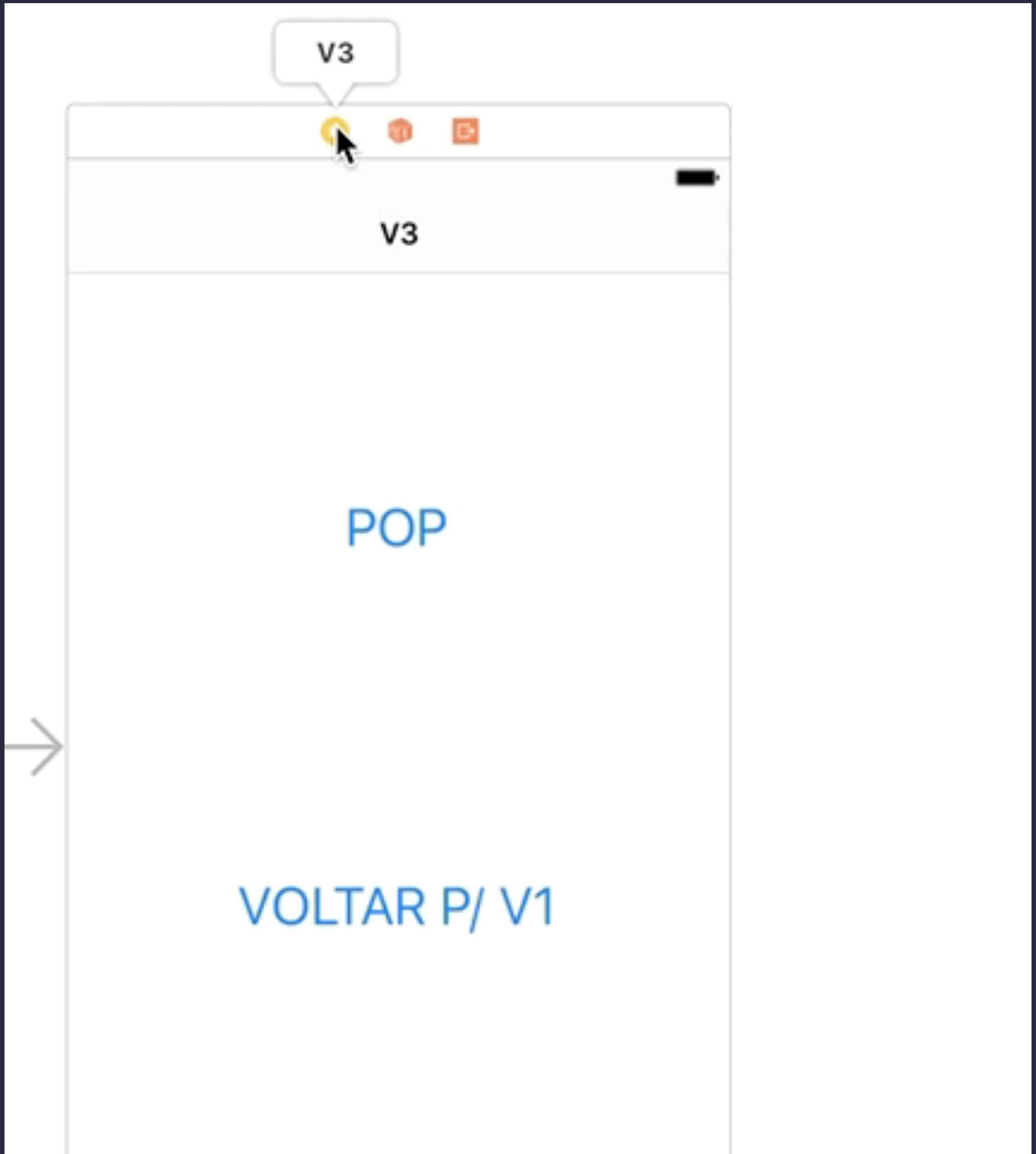
1. Criar action no **VC de destino**
    1. @IBAction func unwindToX(segue:)
2. Criar segue do VC de **origem** para o **Exit**
    2.1. dar um **identificador** para a segue
3. **Chamar** a segue
    1. performSegue(withIdentifier: sender:)

// algoritmo

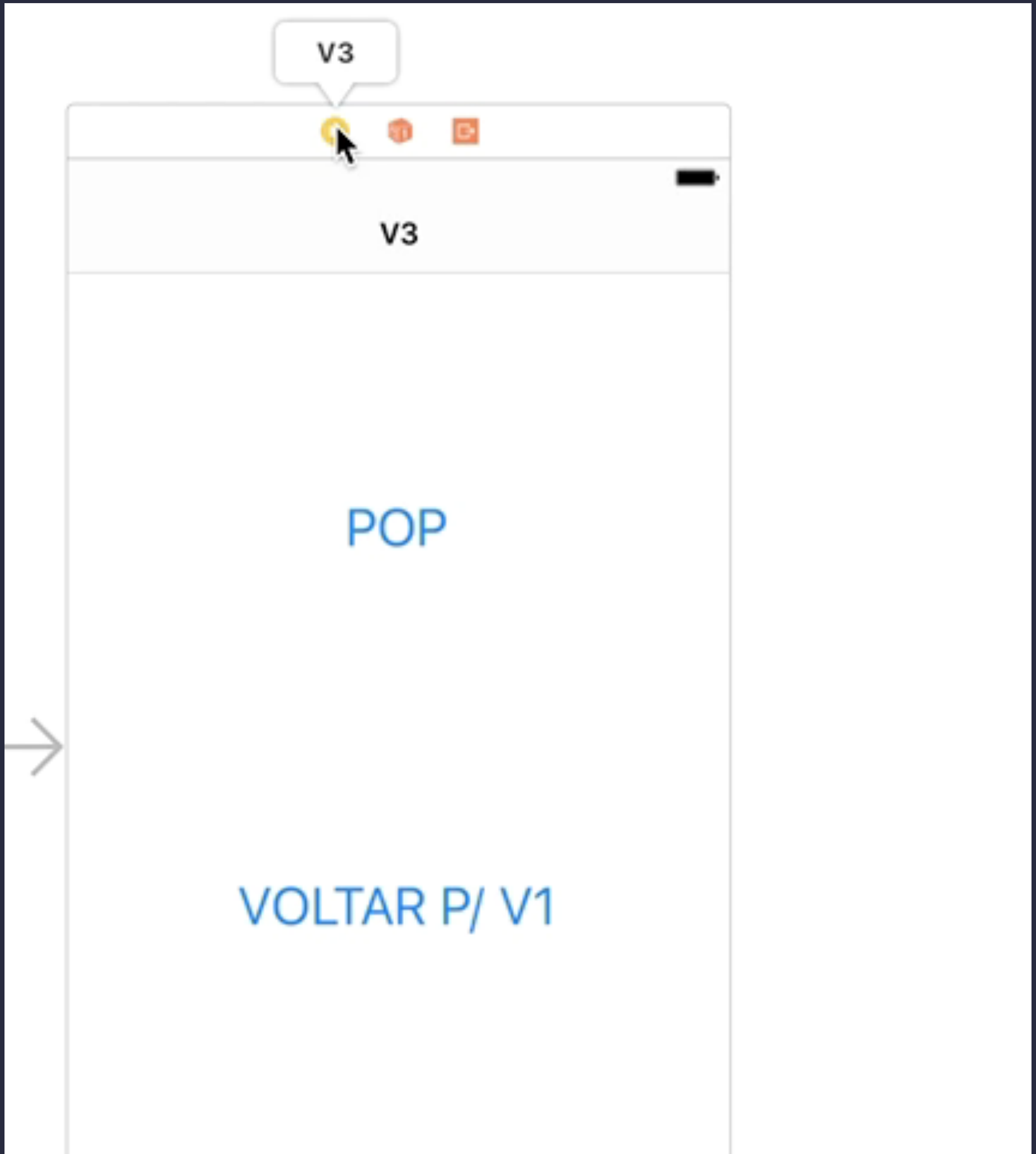

Navigation Controller
Navigation Controller
V1
V1
IR PARA V2
V2
V2
IR PARA V3
V3
V3
POP
VOLTAR P/ V1

```
    // 1. no VC1

@IBAction func unwindToVC1(sender: UIStoryboardSegue)
{ … }
```

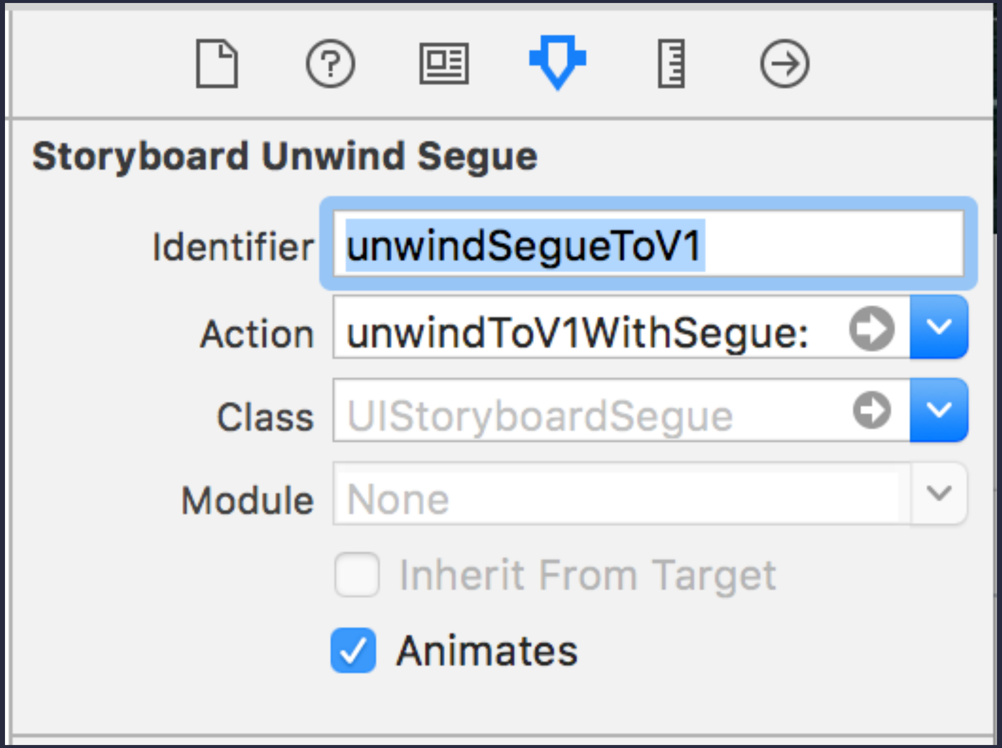
// 2. criando segue



// 2. criando segue



// 2. nomeando segue



Storyboard Unwind Segue

Identifier: unwindSegueToV1

Action: unwindToV1WithSegue:

Class: UIStoryboardSegue

Module: None

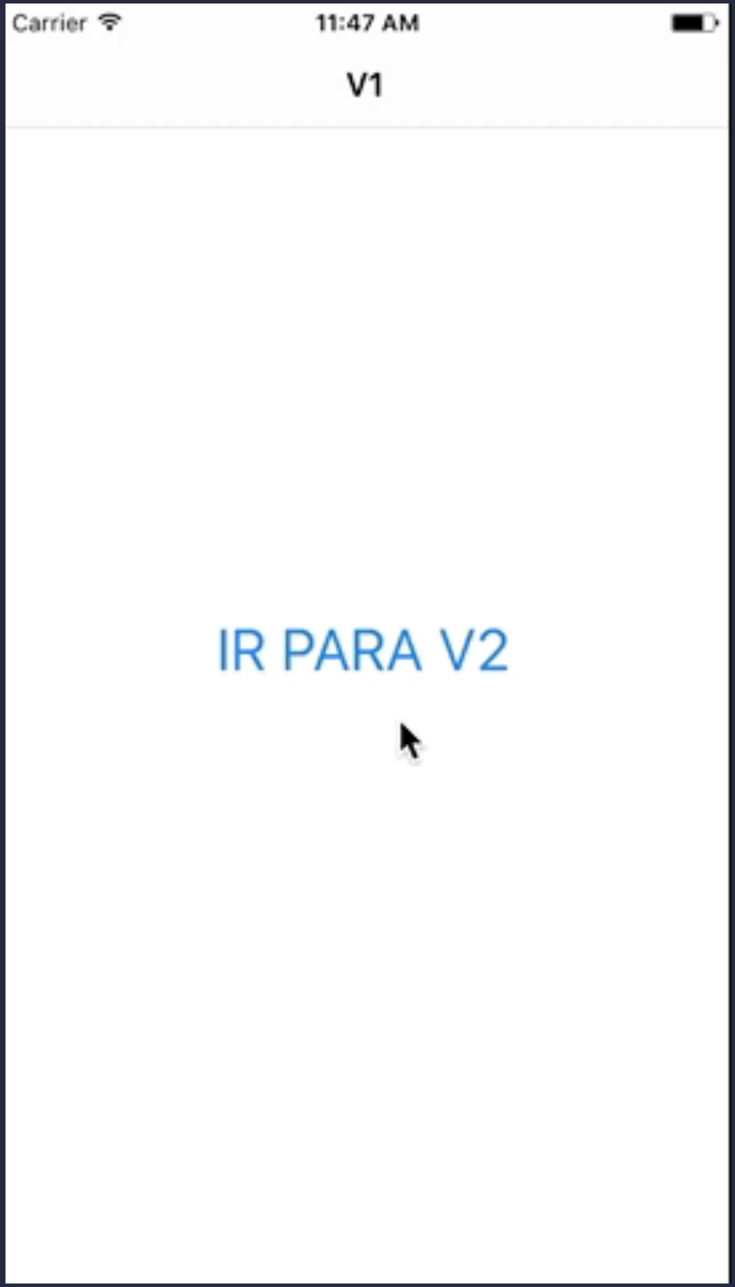
□ Inherit From Target

☑ Animates

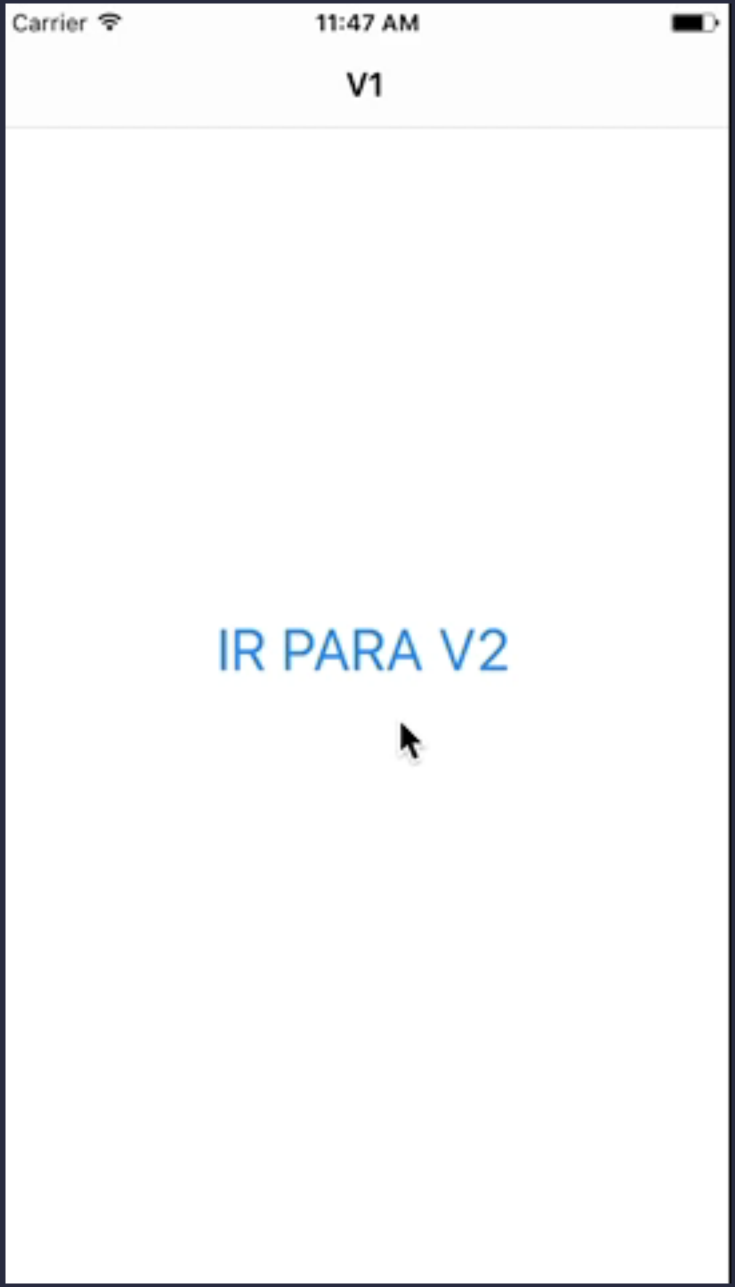
// 3. chamar segue

```
@IBAction func voltarV1(_ sender: Any) {
    self.performSegue(withIdentifier: "unwindSegueToV1", sender: self)
}
```

// 3. chamar segue



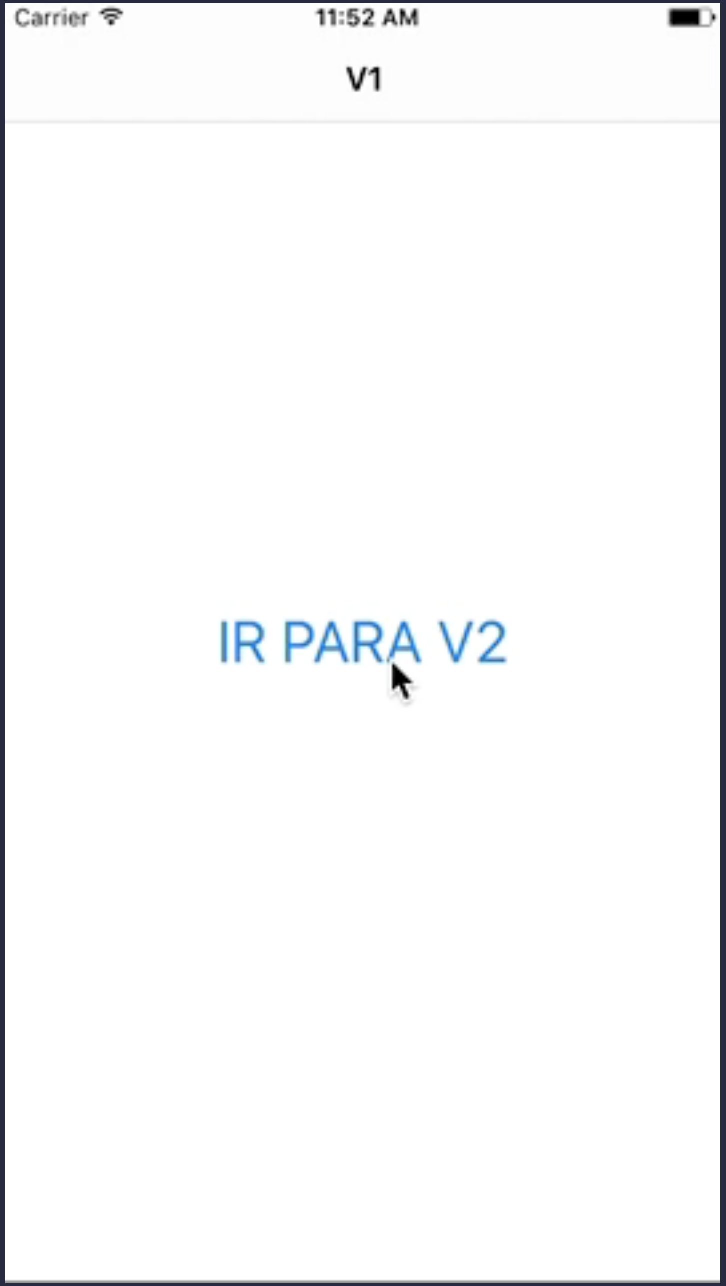
// 3. chamar segue



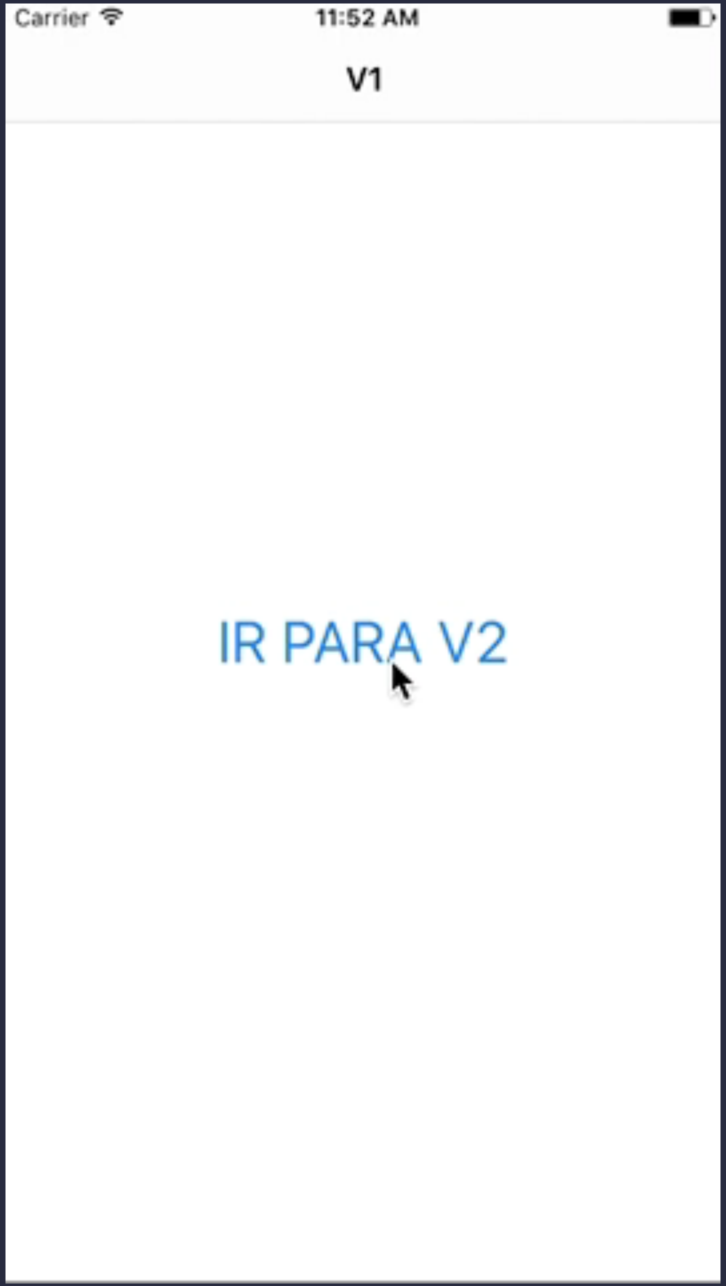
// extra. pop

```
@IBAction func popBtn(_ sender: Any) {
    /* se fosse modal:
    dismiss(animated: true, completion: nil)*/
    self.navigationController?.popViewController(animated: true)
}
```

// extra. pop



// extra. pop



// Persistência

// Core Data

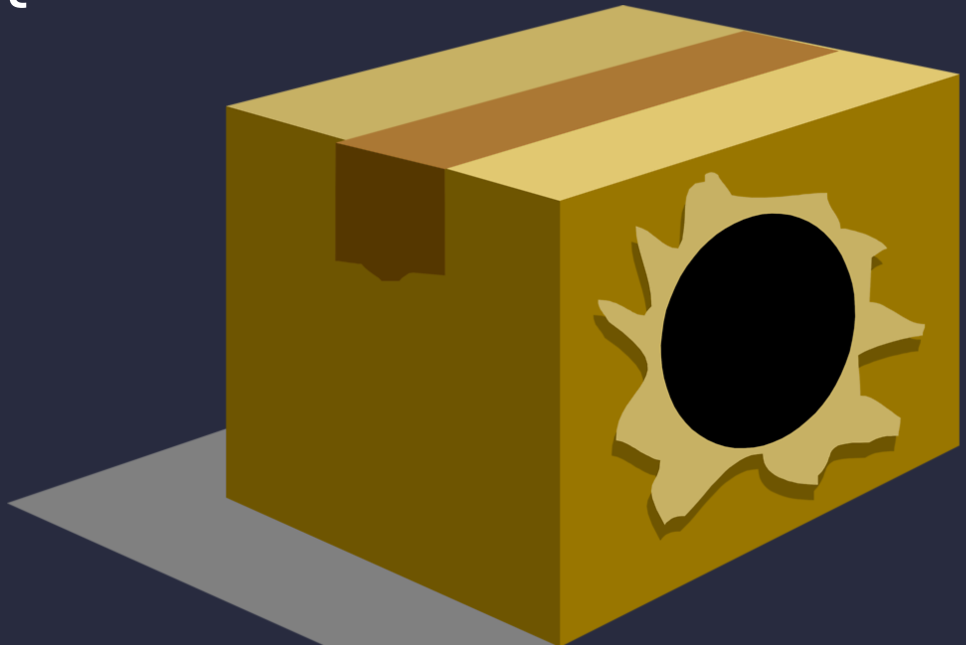
1. Framework de Persistência local
2. Usa a memória do iPhone
3. Usa SQLite
4. Modelar os dados

// Core Data

1. NSPersistentContainer
2. NSManagedObjectContext
3. NSManagedObject

// Core Data

1. NSPersistentContainer
2. NSManagedObjectContext
3. NSManagedObject



// Core Data

Carrier
3:41 PM
Pessoas
+

// Core Data

// algoritmo geral

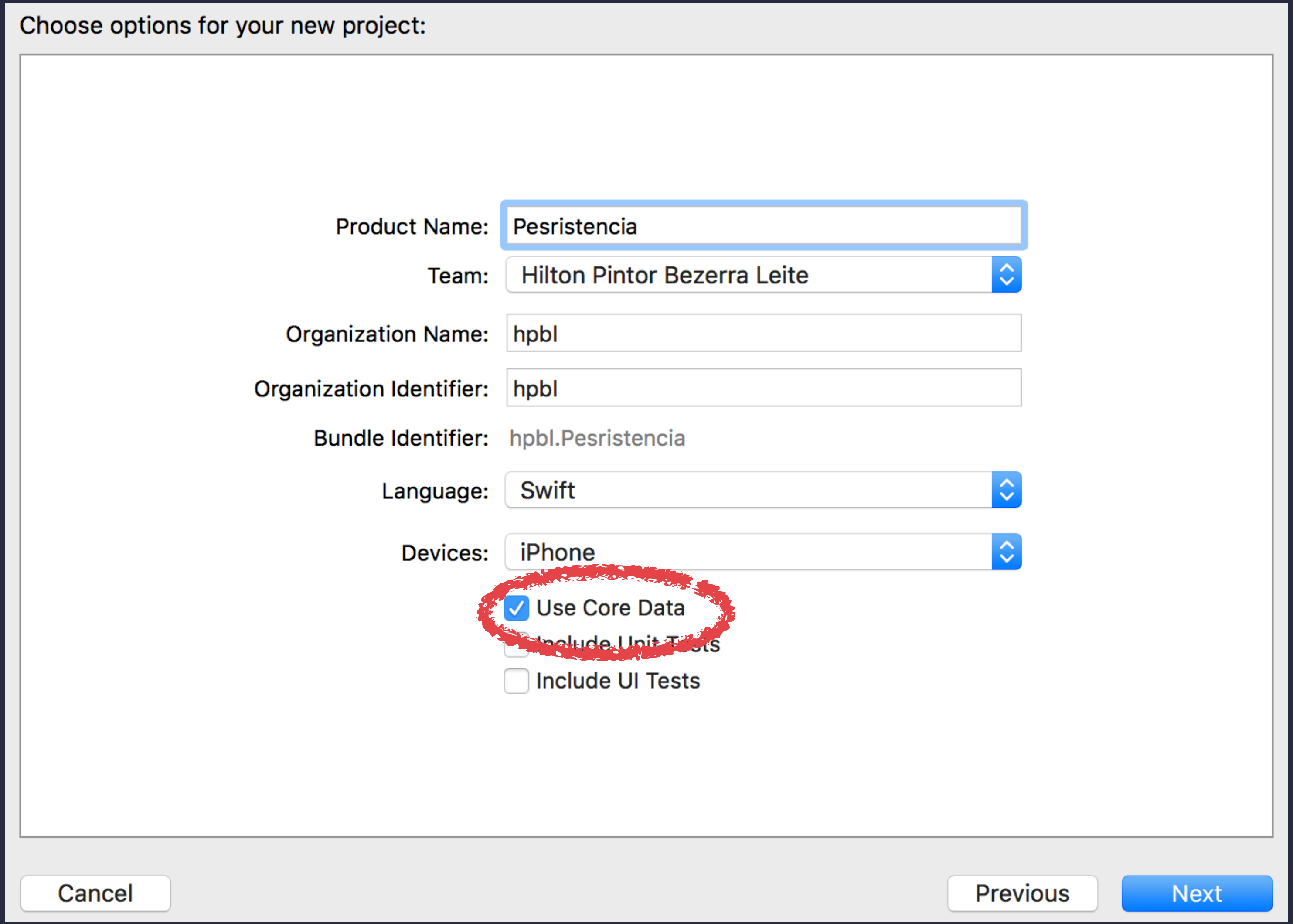
1. Criar projeto com **Core Data**
2. Criar **Entity** e dar **Attributes**
3. Instanciar **App Delegate** para pegar o NSManagedContext
4. Realizar **fetch** dos dados salvos
5. Quando adicionar novos dados, **save**
6. Quando remover dados, **delete**

/*

1. Criar projeto com **Core Data**

*/

// 1. Criar Projeto



Choose options for your new project:

Product Name: Pesristencia

Team: Hilton Pintor Bezerra Leite

Organization Name: hpbl

Organization Identifier: hpbl

Bundle Identifier: hpbl.Pesristencia

Language: Swift

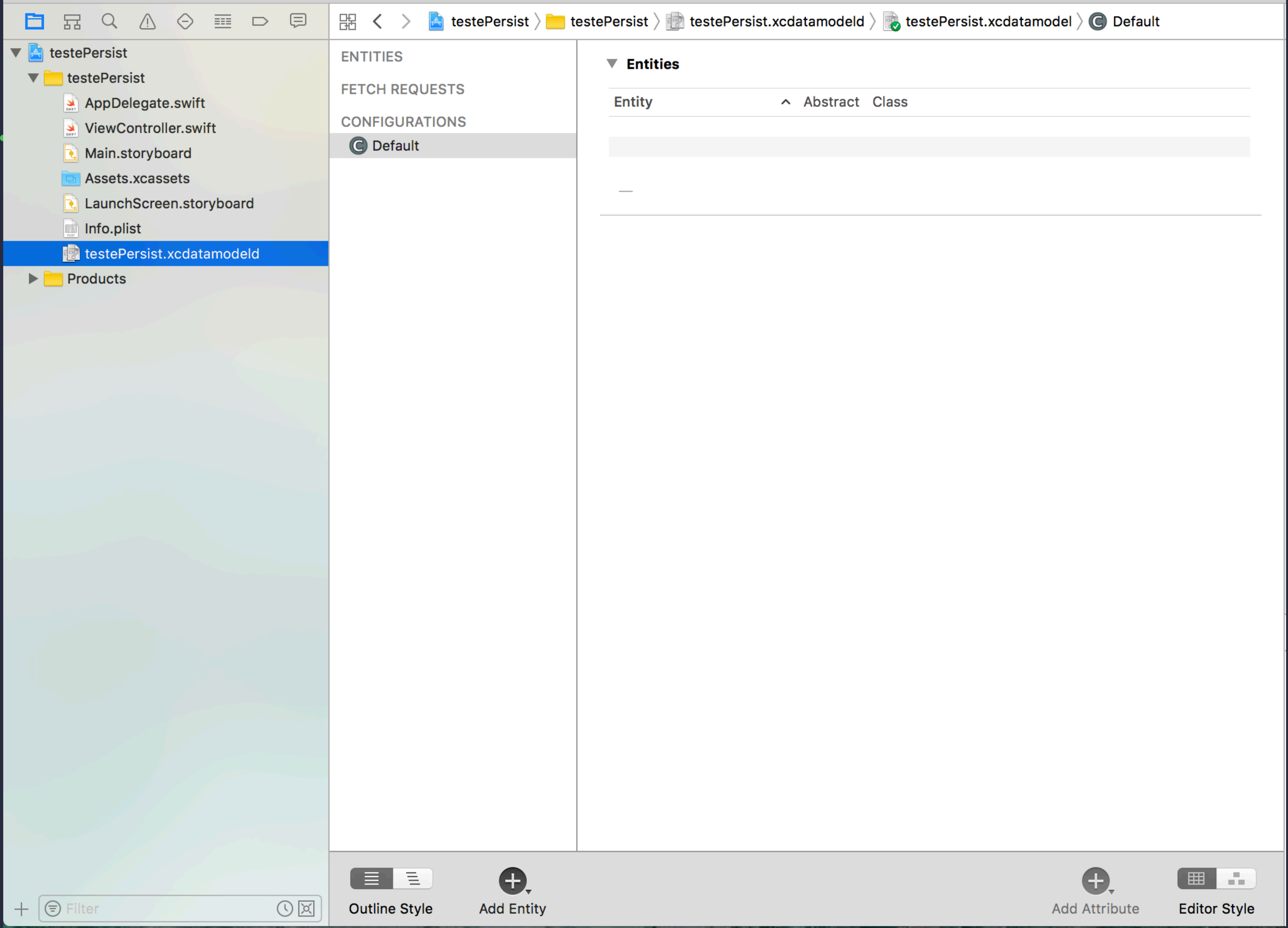
Devices: iPhone

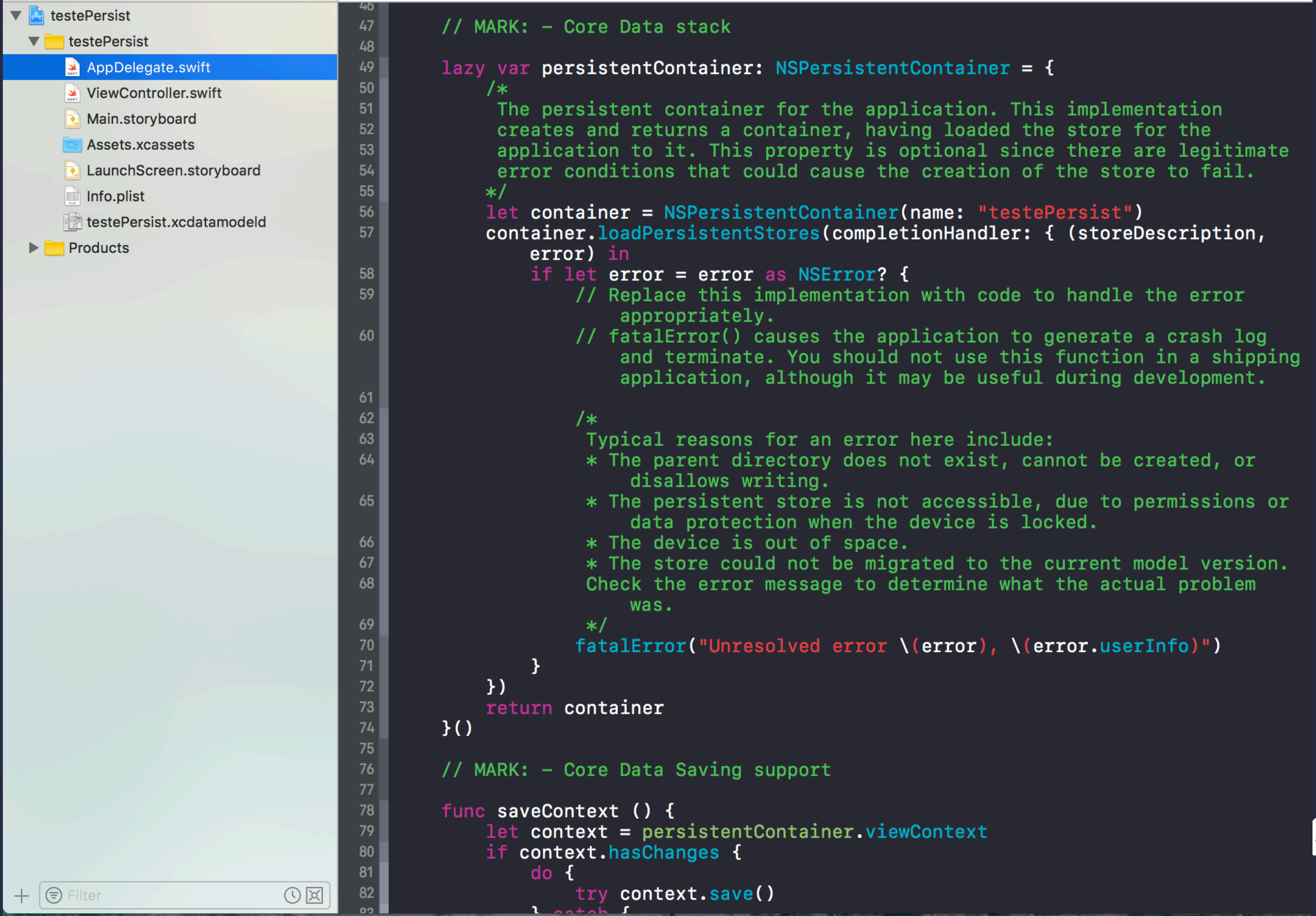
☑ Use Core Data

□ Include Unit Tests

□ Include UI Tests

// 1


testePersist
testePersist
AppDelegate.swift
ViewController.swift
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
testePersist.xcdatamodeld
Products
Filter
testePersist 〉 testePersist 〉 testePersist.xcdatamodeld 〉 testePersist.xcdatamodel 〉 Default
ENTITIES
FETCH REQUESTS
CONFIGURATIONS
Default
Entities
Entity
Abstract
Class
Outline Style
Add Entity
Add Attribute
Editor Style



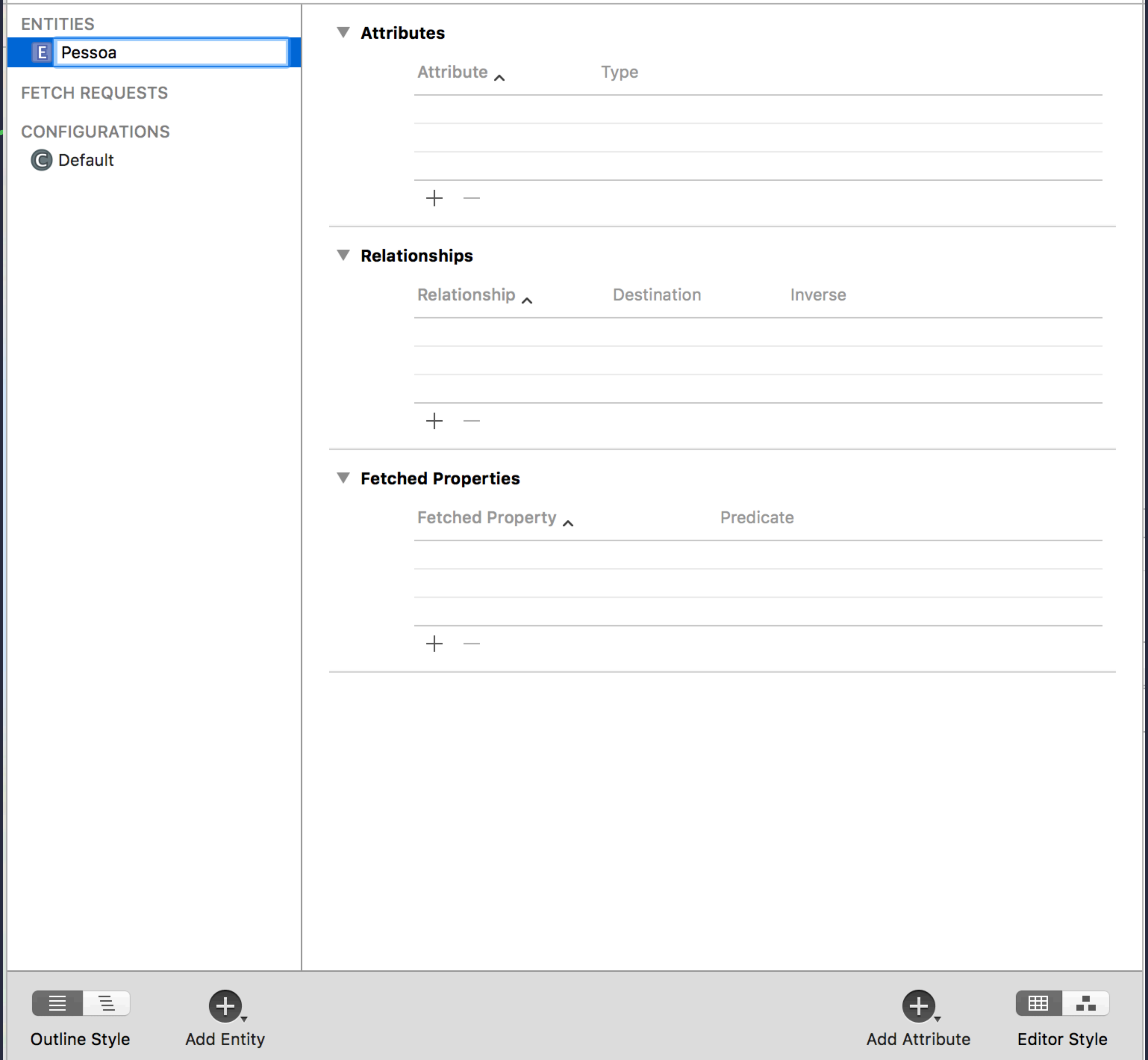
```
    // MARK: - Core Data stack

    lazy var persistentContainer: NSPersistentContainer = {
        /*
         The persistent container for the application. This implementation creates and returns a container, having loaded the store for the application to it. This property is optional since there are legitimate error conditions that could cause the creation of the store to fail.
        */
        let container = NSPersistentContainer(name: "testePersist")
        container.loadPersistentStores(completionHandler: { (storeDescription, error) in
            if let error = error as NSError? {
                // Replace this implementation with code to handle the error appropriately.
                // fatalError() causes the application to generate a crash log and terminate. You should not use this function in a shipping application, although it may be useful during development.

                /*
                 Typical reasons for an error here include:
                 * The parent directory does not exist, cannot be created, or disallows writing.
                 * The persistent store is not accessible, due to permissions or data protection when the device is locked.
                 * The device is out of space.
                 * The store could not be migrated to the current model version.
                 Check the error message to determine what the actual problem was.
                 */
                fatalError("Unresolved error \(error), \(error.userInfo)")
            }
        })
        return container
    }()

    // MARK: - Core Data Saving support

    func saveContext () {
        let context = persistentContainer.viewContext
        if context.hasChanges {
            do {
                try context.save()
            } catch {
```

/*

2. Criar **Entity** e dar **Attributes**

*/

// 2. criar



ENTITIES

E Pessoa

FETCH REQUESTS

CONFIGURATIONS

C Default

Attributes

Attribute	Type

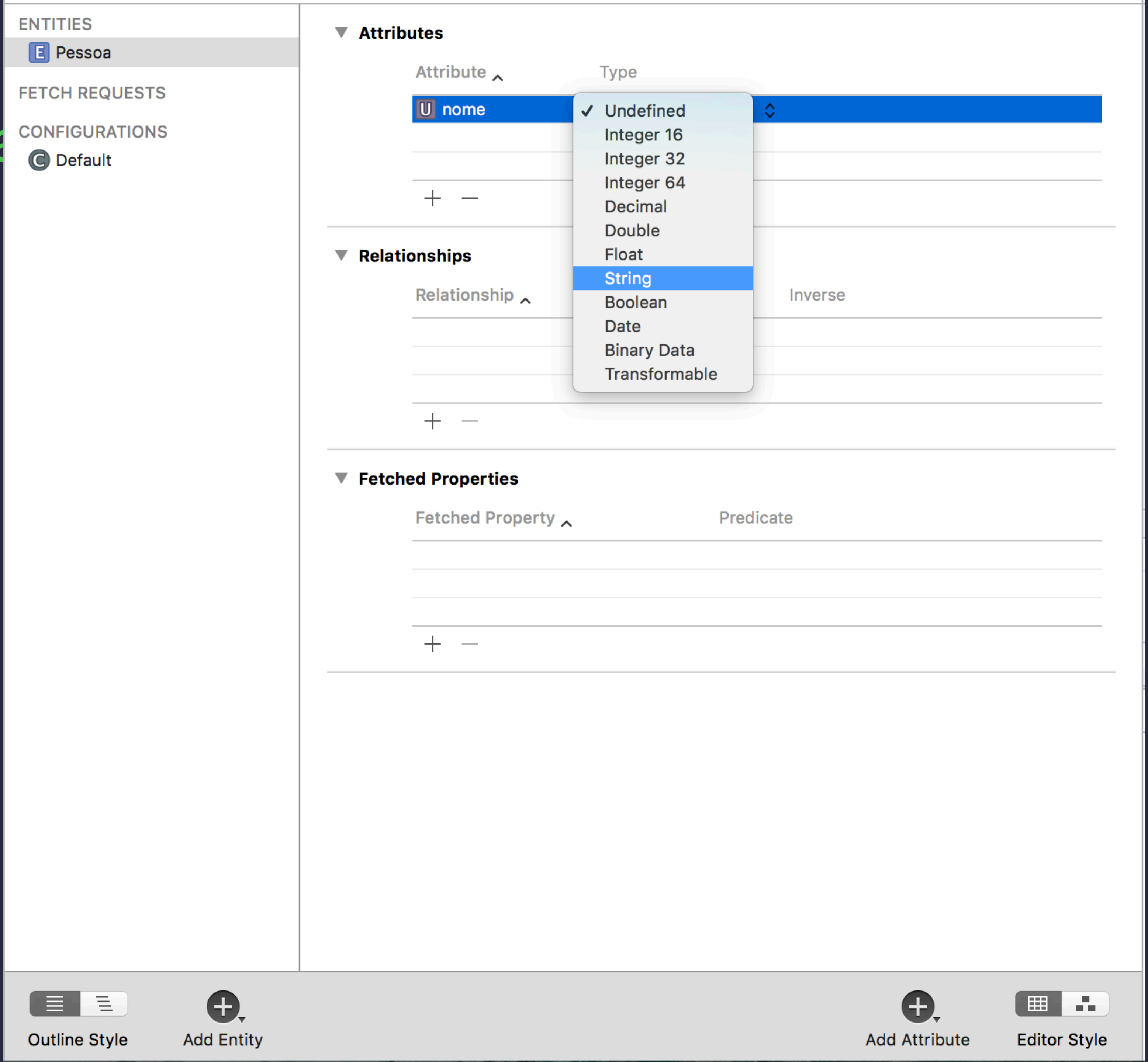
Relationships

Relationship	Destination	Inverse

Fetched Properties

Fetched Property	Predicate

// 2. dar a


ENTITIES
E Pessoa
FETCH REQUESTS
CONFIGURATIONS
C Default
Attributes
Attribute
Type
U nome
✓ Undefined
Integer 16
Integer 32
Integer 64
Decimal
Double
Float
String
Boolean
Date
Binary Data
Transformable
Relationships
Relationship
Inverse
Fetched Properties
Fetched Property
Predicate
Outline Style
Add Entity
Add Attribute
Editor Style

/*

3. Instanciar **App Delegate** para pegar o NSManagedContext

*/

// 3. instanciar app delegate e context

```swift
class ViewController: UIViewController {
    var appDelegate: AppDelegate?
    var managedContext: NSManagedObjectContext?
}
```

// 3. instanciar app delegate e context

```
class ViewController: UIViewController {
    var appDelegate: AppDelegate?
    var managedContext: NSManagedObjectContext?

    override func viewDidLoad() {
        super.viewDidLoad()
        // ...

        self.appDelegate = UIApplication.shared.delegate as? AppDelegate

        self.managedContext = appDelegate?.persistentContainer.viewContext
    }
}
```

/*

4. Realizar **fetch** dos dados salvos

*/

// 4. fetch

```
class ViewController: UIViewController {
    var appDelegate: AppDelegate?
    var managedContext: NSManagedObjectContext?
    var pessoas: [NSManagedObject] = []

    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)

        let fetchRequest = NSFetchRequest<NSManagedObject>(entityName: "Pessoa")

        do {
            try self.pessoas = (self.managedContext?.fetch(fetchRequest))!
        } catch let error as NSError {
            print("erro na hora de pedir. \(error), \(error.userInfo)")
        }
    }
}
```

/*

5. Quando adicionar novos dados, **save**

*/

// 5. save

```
class ViewController: UIViewController {
    var appDelegate: AppDelegate?
    var managedContext: NSManagedObjectContext?
    var pessoas: [NSManagedObject] = []

    func save(novoNome: String) {
        let entity = NSEntityDescription.entity(forEntityName: "Pessoa", in: managedContext!)

        let pessoa = NSManagedObject(entity: entity!, insertInto: managedContext)

        pessoa.setValue(novoNome, forKey: "nome")

        do {
            try managedContext?.save()
            self.pessoas.append(pessoa)
        } catch let error as NSError {
            print("erro na hora de salvar. \(error), \(error.userInfo)")
        }
    }
}
```

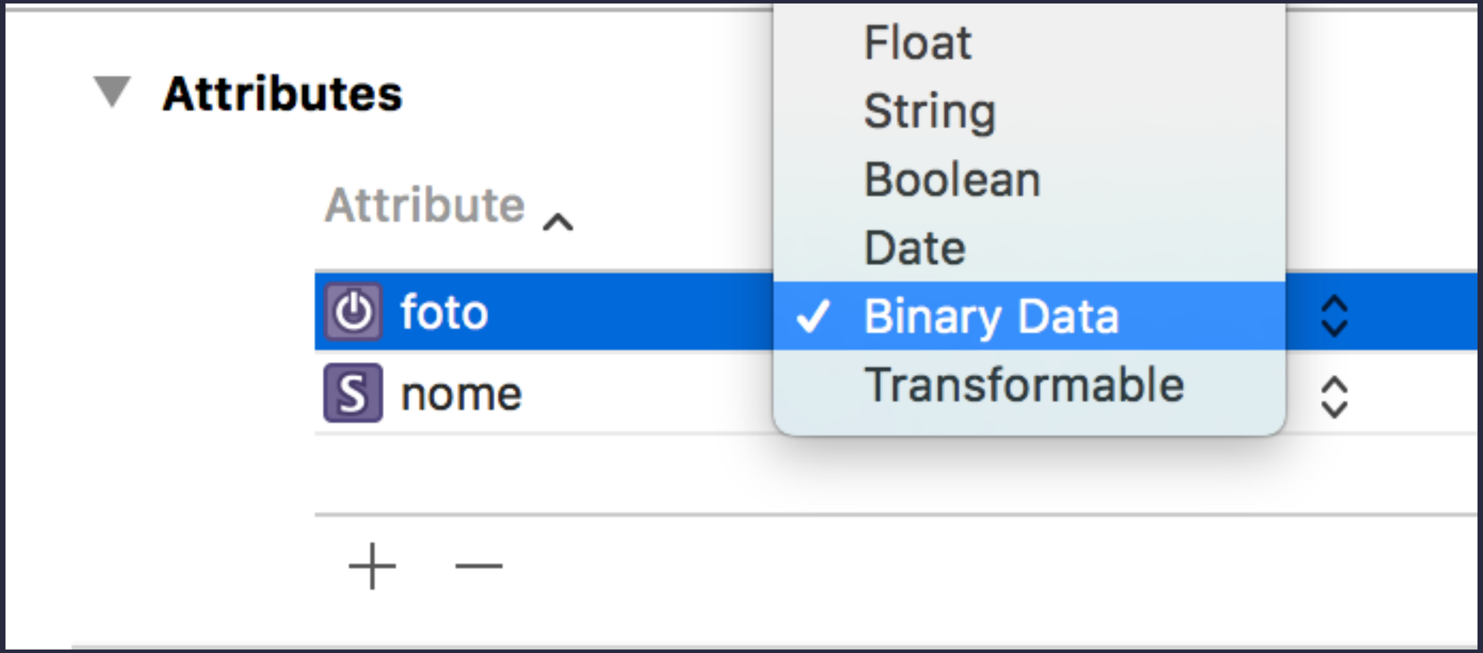
/*

5. Quando remover dados, **delete**

*/

```
  extension ViewController: UITableViewDataSource {
// 5. save
      func tableView(_ tableView: UITableView,
                        commit editingStyle: UITableViewCellEditingStyle,
                        forRowAt indexPath: IndexPath) {

          if editingStyle == .delete {
              let pessoa = self.pessoas[indexPath.row]
              self.managedContext?.delete(pessoa)
              self.appDelegate?.saveContext()

              let fetchRequest = NSFetchRequest<NSManagedObject>(entityName: "Pessoa")

              do {
                  try self.pessoas = (self.managedContext?.fetch(fetchRequest))!
                  tableView.reloadData()
              } catch {
                  print("Fetching Failed")
          }
        }
     }
  }
```

// persistindo Imagens



Attributes

Attribute

foto ✓ Binary Data

nome

// save Imagens

```
func save(novoNome: String) {
    let entity = NSEntityDescription.entity(forEntityName: "Pessoa", in: managedContext!)

    let pessoa = NSManagedObject(entity: entity!, insertInto: managedContext)

    let img = #imageLiteral(resourceName: "diego")
    let imgData = UIImageJPEGRepresentation(img, 1)

    pessoa.setValue(imgData, forKey: "foto")

    pessoa.setValue(novoNome, forKey: "nome")

    do {
        try managedContext?.save()
        self.pessoas.append(pessoa)
    } catch let error as NSError {
        print("erro na hora de salvar. \(error), \(error.userInfo)")
    }
}
```

// fetch Imagens

```
func tableView(_ tableView: UITableView,
                        cellForRowAt indexPath: IndexPath) -> UITableViewCell {

    let cell = tableView.dequeueReusableCell(withIdentifier: “Cell",
                                                        for: indexPath)

    let pessoa = pessoas[indexPath.row]

    cell.textLabel?.text = pessoa.value(forKey: "nome") as? String

    guard let imgData = pessoa.value(forKey: "foto") as? Data,
          let image = UIImage(data: imgData) else {
        return cell
    }

    cell.imageView?.image = image

    return cell
}
```

/*

Como adicionar Core Data a um projeto existente?

*/

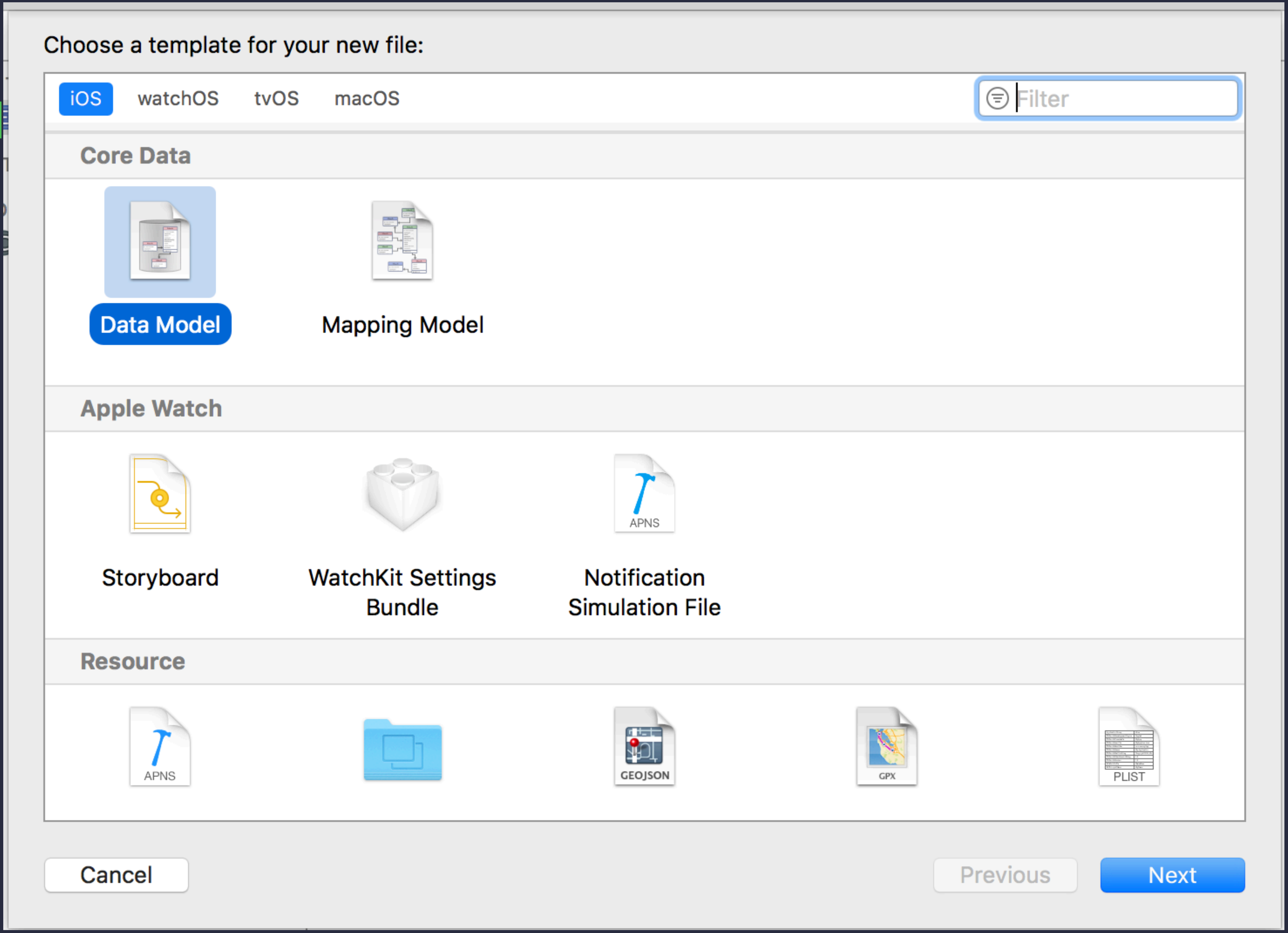
// algoritmo

1. Adicionar arquivo **Data Model** (.xcdatamodeld**)**
2. Adicionar código ao **App Delegate**

    2.1. Mudar `let container = NSPersistentContainer(name: “nomeDoArquivo”)`

3. Adaptar o código para usar Core Data

// 1



Choose a template for your new file:

iOS watchOS tvOS macOS

Filter

Core Data

Data Model

Mapping Model

Apple Watch

Storyboard

WatchKit Settings Bundle

Notification Simulation File

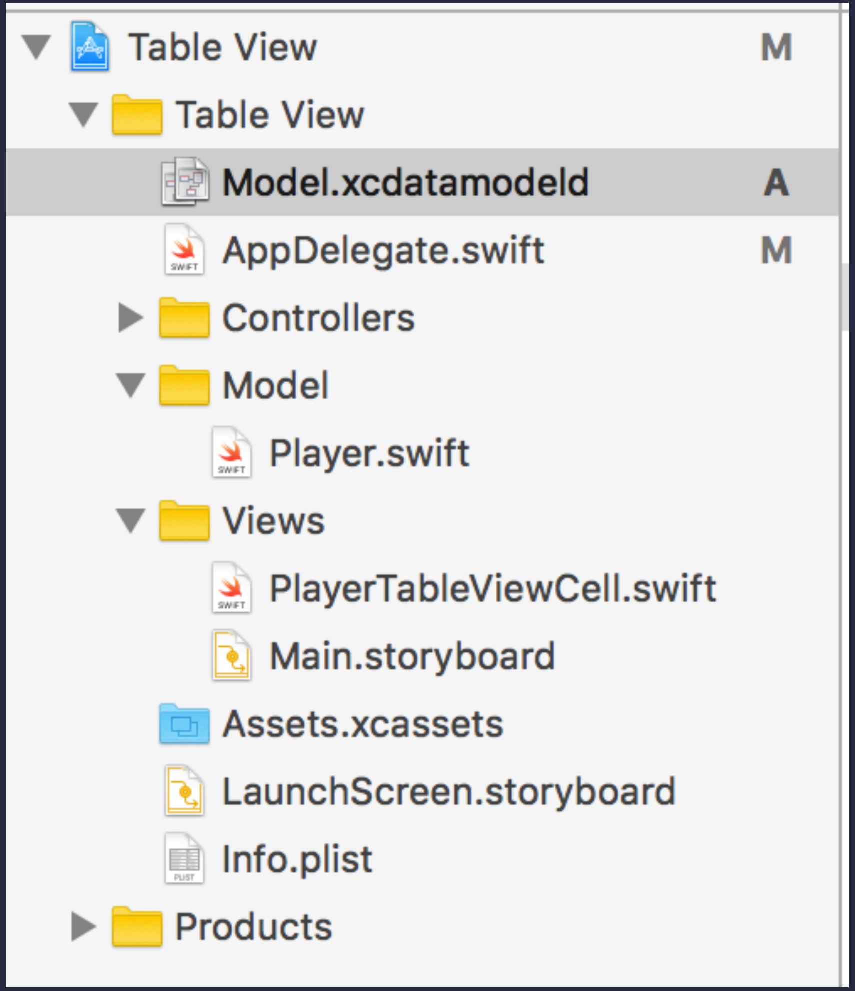
Resource

Cancel

Previous

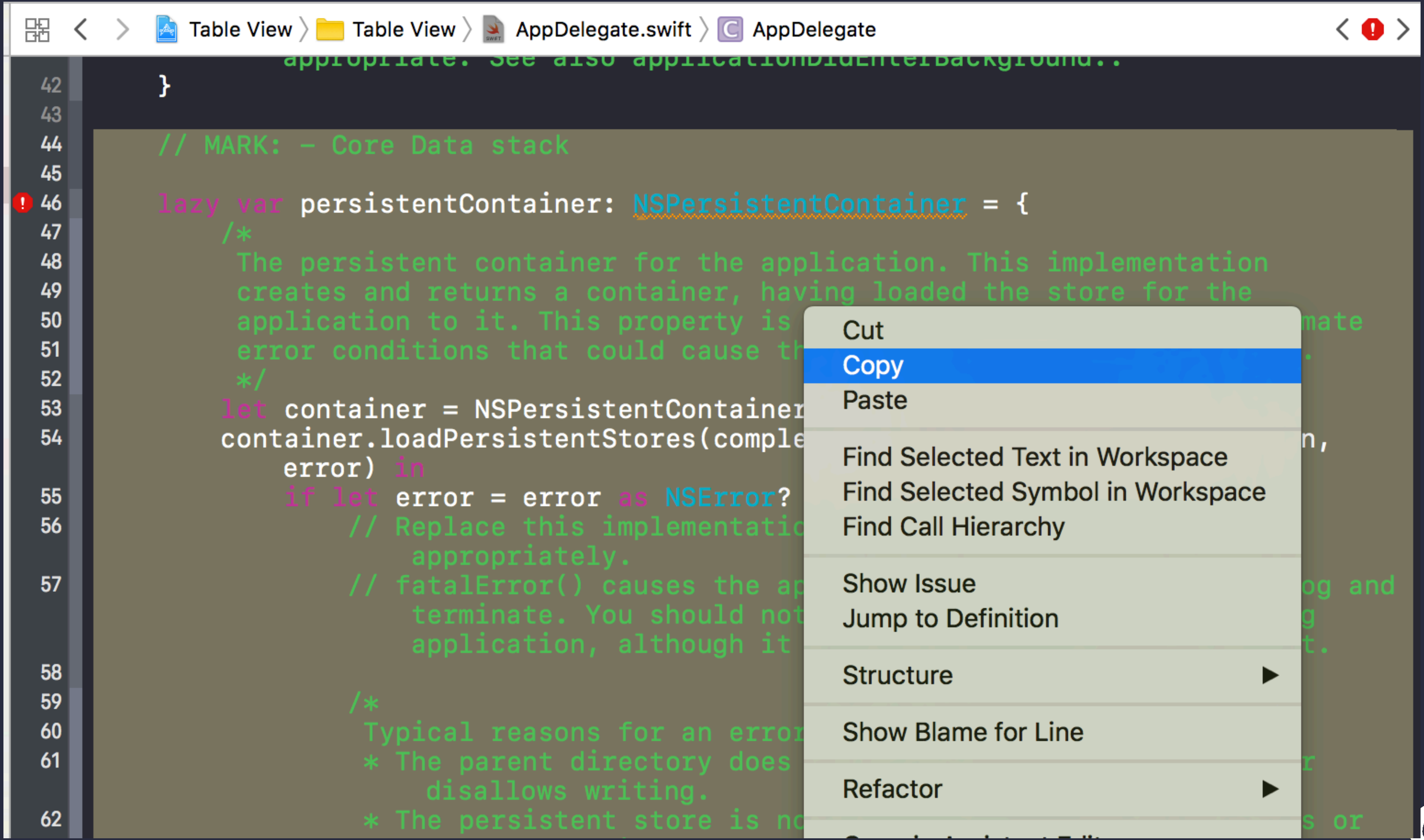
Next

// 1. Adicionar arquivo **.xcdatamodeld**



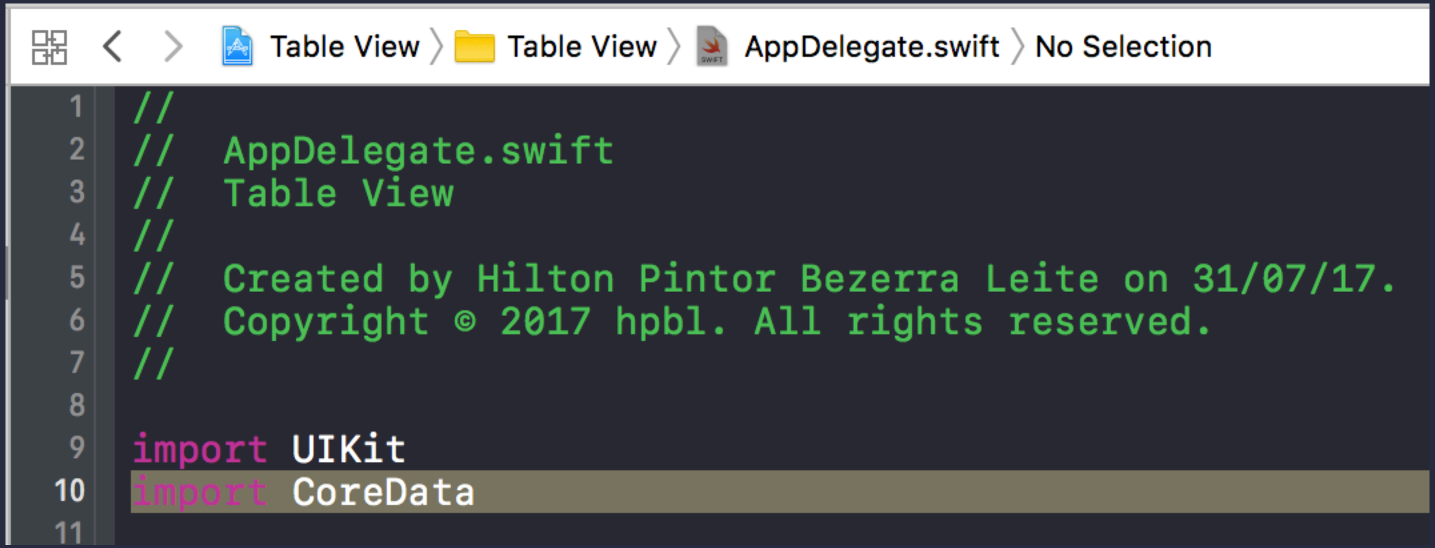
```
▼ Table View                         M
  ▼ Table View
      Model.xcdatamodeld             A
      AppDelegate.swift              M
    ▶ Controllers
    ▼ Model
        Player.swift
    ▼ Views
        PlayerTableViewCell.swift
        Main.storyboard
      Assets.xcassets
      LaunchScreen.storyboard
      Info.plist
  ▶ Products
```

// 2. Adicionar código ao **App Delegate**



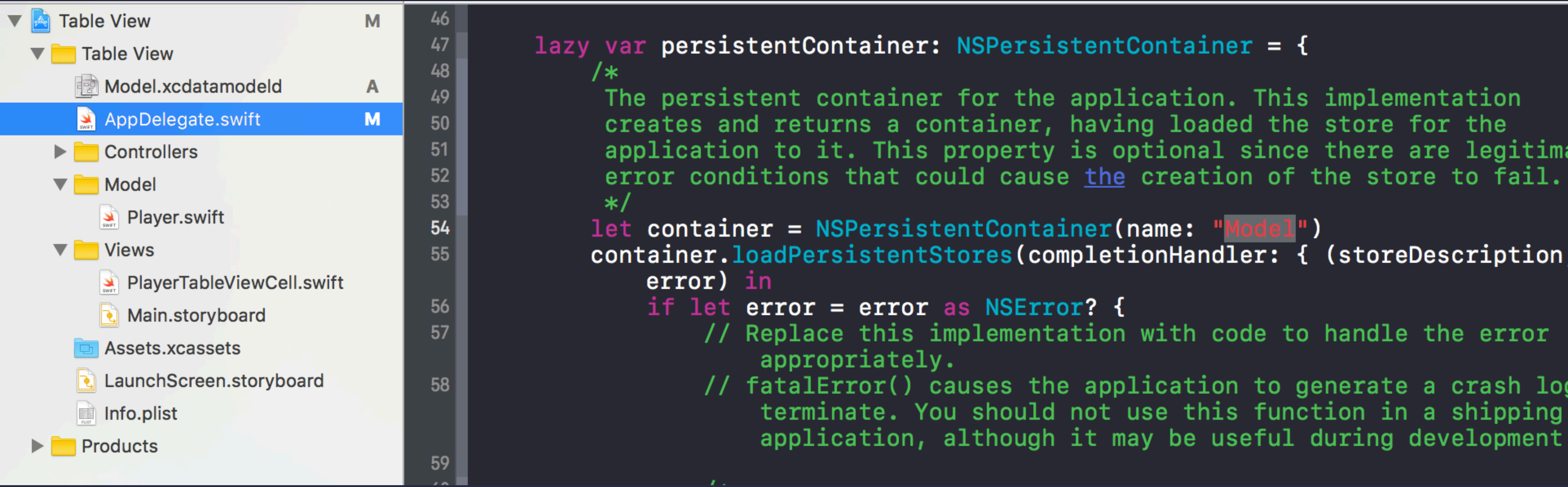
```
        appropriate. See also applicationDidEnterBackground:.
    }

    // MARK: - Core Data stack

    lazy var persistentContainer: NSPersistentContainer = {
        /*
         The persistent container for the application. This implementation
         creates and returns a container, having loaded the store for the
         application to it. This property is                              mate
         error conditions that could cause th                             .
        */
        let container = NSPersistentContainer
        container.loadPersistentStores(comple                             n,
            error) in
            if let error = error as NSError?
                // Replace this implementatio
                    appropriately.
                // fatalError() causes the ap                              og and
                    terminate. You should not                              g
                    application, although it                               t.

                /*
                 Typical reasons for an error
                 * The parent directory does                               r
                    disallows writing.
                 * The persistent store is no                              s or
```

// 2. Importar Core Data



```
//
//  AppDelegate.swift
//  Table View
//
//  Created by Hilton Pintor Bezerra Leite on 31/07/17.
//  Copyright © 2017 hpbl. All rights reserved.
//

import UIKit
import CoreData
```

// 3. Mudar nome do container



```
    lazy var persistentContainer: NSPersistentContainer = {
        /*
         The persistent container for the application. This implementation
         creates and returns a container, having loaded the store for the
         application to it. This property is optional since there are legitima
         error conditions that could cause the creation of the store to fail.
         */
        let container = NSPersistentContainer(name: "Model")
        container.loadPersistentStores(completionHandler: { (storeDescription
            error) in
            if let error = error as NSError? {
                // Replace this implementation with code to handle the error
                   appropriately.
                // fatalError() causes the application to generate a crash lo
                   terminate. You should not use this function in a shipping
                   application, although it may be useful during development
```

// Exercício

// Exercício 14

Lista de coisas III

1. Adicione persistência local usando **Core Data** ao seu app
2. Dados **adicionados** devem ser mantidos
3. **Remoções** devem ser mantidas

// Extra

4. **Edições** devem ser mantidas

DÚVIDAS

?

citi