
cit


Hilton **Pintor**
Desenvolvedor (iOS/tvOS/watchOS)
hiltonpintor@gmail.com
citi

// Aula 05

// Recapitulando

// Delegate

// Delegate

1. Definição do protocolo
2. Implementar protocolo em uma classe
3. adicionar propriedade delegate
4. chamar métodos do delegate

```swift
// delegate

import UIKit

class ViewController: UIViewController {
    @IBOutlet weak var colorTextField: UITextField!

    @IBOutlet weak var colorBtn: UIButton!

    @IBAction func changeColor(_ sender: Any) {
        var novaCor: UIColor

        // ...
}
```

```swift
// delegate
extension ViewController: UITextFieldDelegate {



}
```

```swift
// delegate
extension ViewController: UITextFieldDelegate {

    func textField(_ textField: UITextField,
                   shouldChangeCharactersIn range: NSRange,
                   replacementString string: String) -> Bool {

        guard let texto = textField.text else {
            return true
        }

        if texto != ""{
            self.colorBtn.isEnabled = true
        }
        return true
    }
}
```

```swift
// delegate

class ViewController: UIViewController {
    @IBOutlet weak var colorTextField: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()


    }
}
```

```swift
// delegate

class ViewController: UIViewController {
    @IBOutlet weak var colorTextField: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()

        self.colorTextField.delegate = self
    }
}
```

// delegate

1. UITextFieldDelegate
2. ViewController: UITextFieldDelegate
3. self.colorTextField.delegate = self
4. chamar métodos do delegate (responsabilidade do Text Field

```
// delegate
public protocol UITextFieldDelegate : NSObjectProtocol {


    @available(iOS 2.0, *)
    optional public func textFieldShouldBeginEditing(_ textField: UITextField) -> Bool // return NO to disallow editing.

    @available(iOS 2.0, *)
    optional public func textFieldDidBeginEditing(_ textField: UITextField) // became first responder

    @available(iOS 2.0, *)
    optional public func textFieldShouldEndEditing(_ textField: UITextField) -> Bool // return YES to allow editing to stop and to resign first responder status. NO to disallow the editing session to end

    @available(iOS 2.0, *)
    optional public func textFieldDidEndEditing(_ textField: UITextField) // may be called if forced even if shouldEndEditing returns NO (e.g. view removed from window) or endEditing:YES called

    @available(iOS 10.0, *)
    optional public func textFieldDidEndEditing(_ textField: UITextField, reason: UITextFieldDidEndEditingReason) // if implemented, called in place of textFieldDidEndEditing:

// ...
}
```

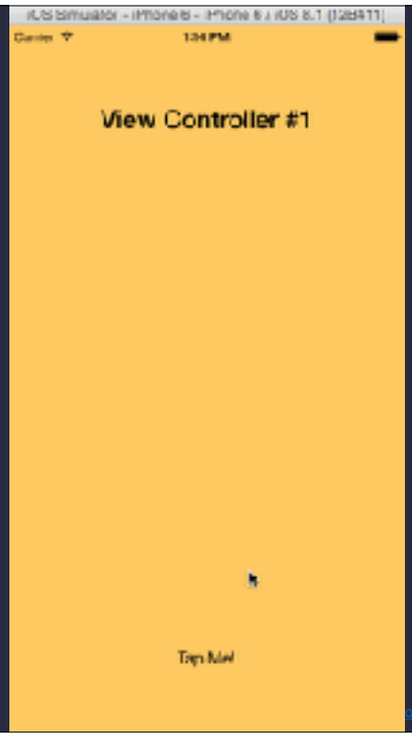
// Navegação

// navegação

Trocar de tela

Trocar de View

Trocar de View Controller



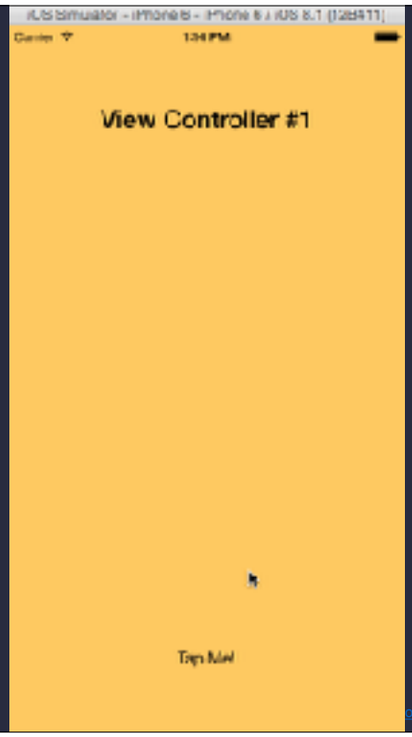
om/custom-segue-animations/

// navegação

Trocar de tela

Trocar de View

Trocar de View Controller



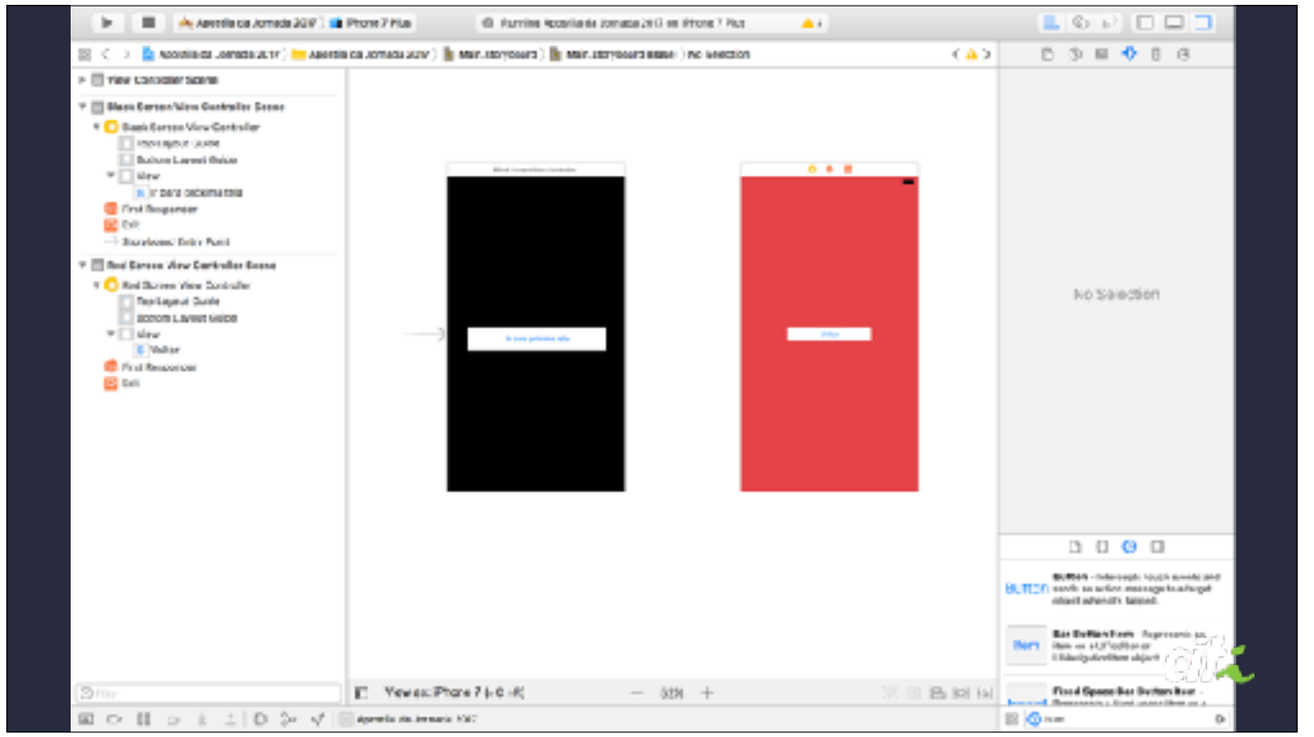
om/custom-segue-animations/

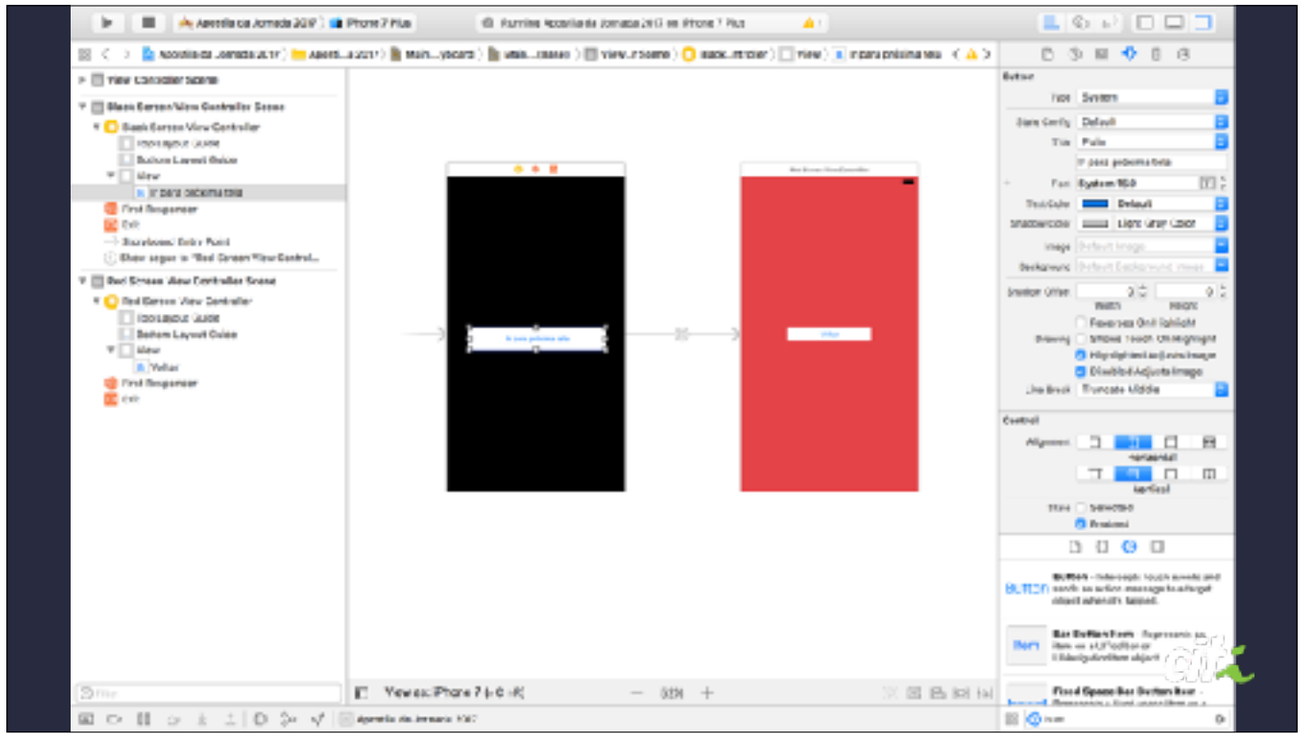
// Segues

// Segues

- Definem o fluxo de **navegação** do app
- **Transição** entre ViewControllers do StoryBoard
- **Início**: button, table row, or gesture recognizer
- **Destino**: ViewController a ser mostrado

https://goo.gl/bRnvzY


No Selection

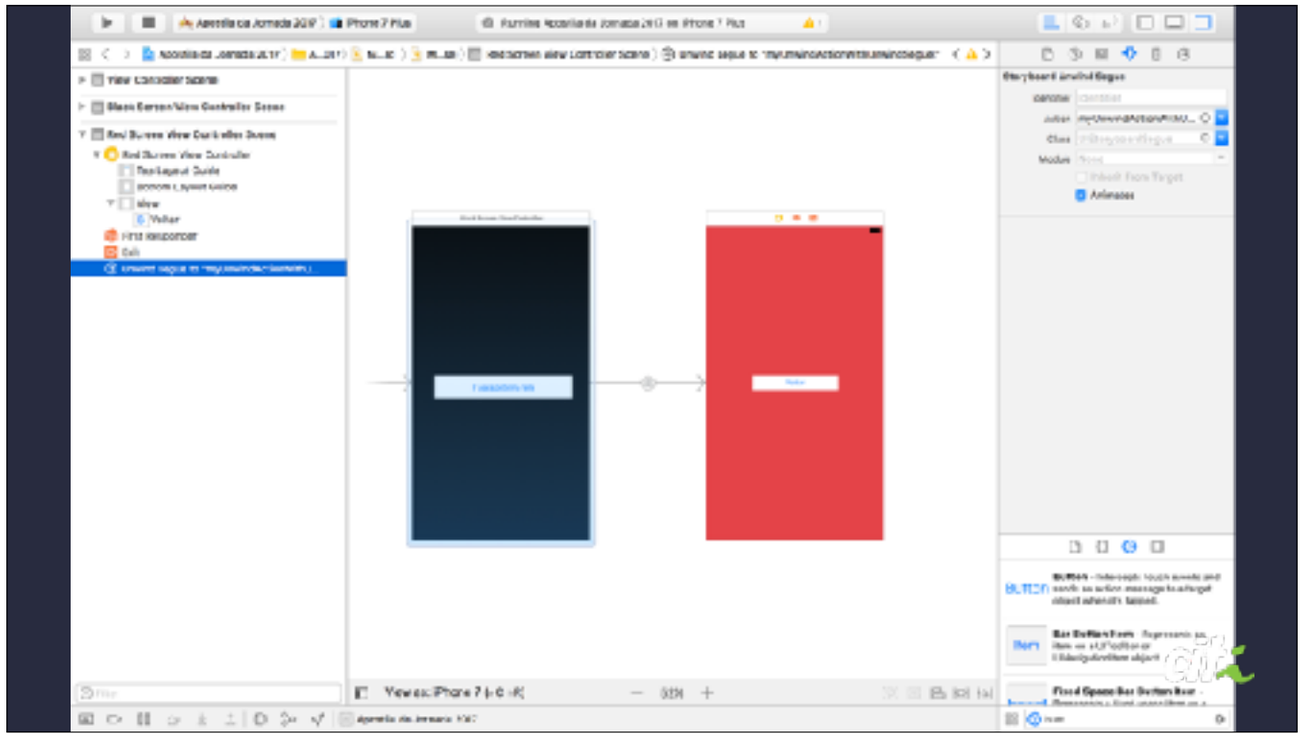


// Segues

- Apresentação Modal

// Segues

- Apresentação Modal



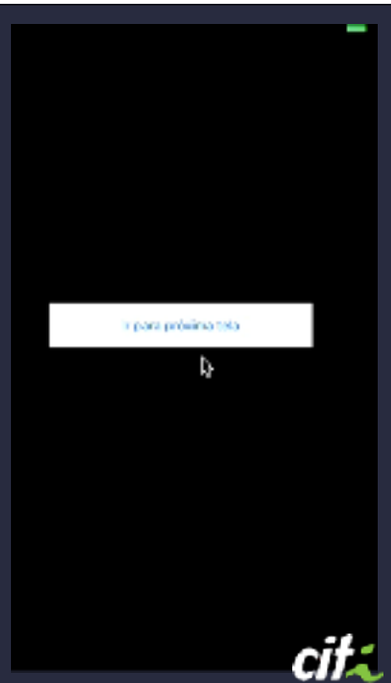
You must define an unwind action method in one of your view controllers before trying to create the corresponding unwind segue in Interface Builder. The presence of that method is required and tells Interface Builder that there is a valid target for the unwind segue.

Use the implementation of your unwind action method to perform any tasks that are specific to your app. You do not need to dismiss any view controllers involved in the segue yourself; UIKit does that for you. Instead, use the segue object to fetch the view controller being dismissed so that you can retrieve data from it. You can also use the unwind action to update the current view controller before the unwind segue finishes.

// Segues

- Unwind

```
@IBAction func myUnwindAction(unwindSegue: UIStoryboardSegue)
```



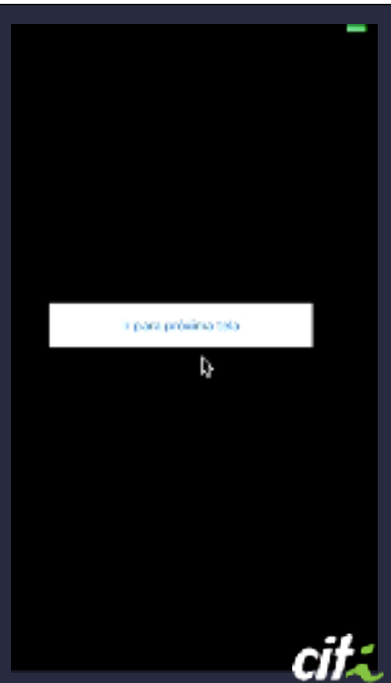
You must define an unwind action method in one of your view controllers before trying to create the corresponding unwind segue in Interface Builder. The presence of that method is required and tells Interface Builder that there is a valid target for the unwind segue.

Use the implementation of your unwind action method to perform any tasks that are specific to your app. You do not need to dismiss any view controllers involved in the segue yourself; UIKit does that for you. Instead, use the segue object to fetch the view controller being dismissed so that you can retrieve data from it. You can also use the unwind action to update the current view controller before the unwind segue finishes.

// Segues

- Unwind

```
@IBAction func myUnwindAction(unwindSegue: UIStoryboardSegue)
```



You must define an unwind action method in one of your view controllers before trying to create the corresponding unwind segue in Interface Builder. The presence of that method is required and tells Interface Builder that there is a valid target for the unwind segue.

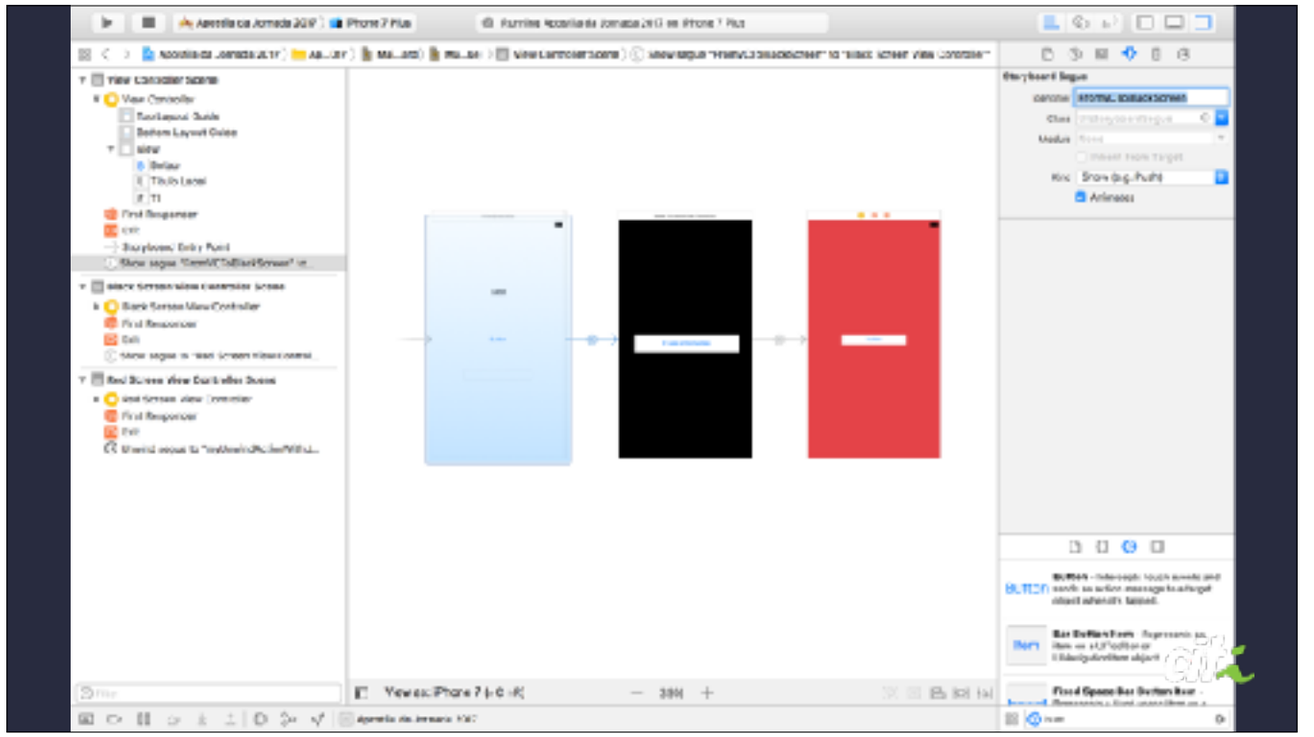
Use the implementation of your unwind action method to perform any tasks that are specific to your app. You do not need to dismiss any view controllers involved in the segue yourself; UIKit does that for you. Instead, use the segue object to fetch the view controller being dismissed so that you can retrieve data from it. You can also use the unwind action to update the current view controller before the unwind segue finishes.

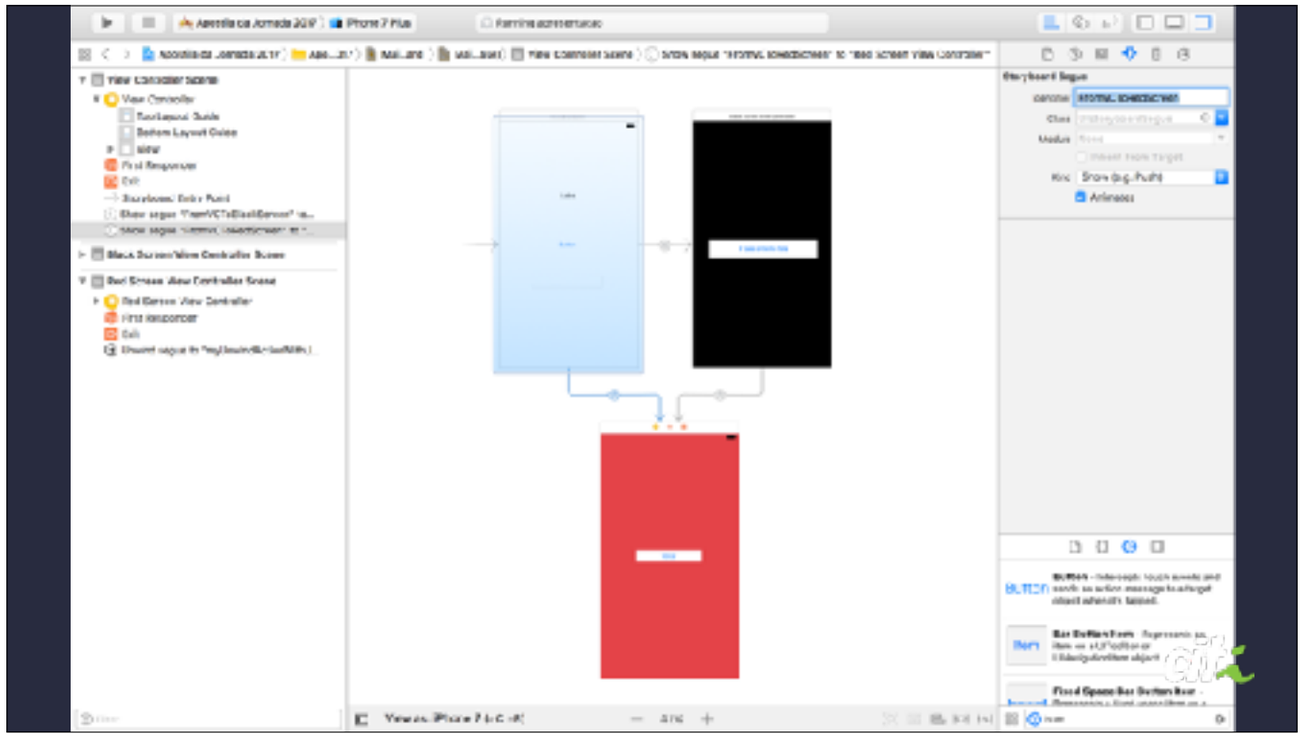
// Chamando por Código

// Segues programáticas

Às vezes é necessário **chamar** uma segue pelo código

https://goo.gl/bRnvzY





// chamando segue

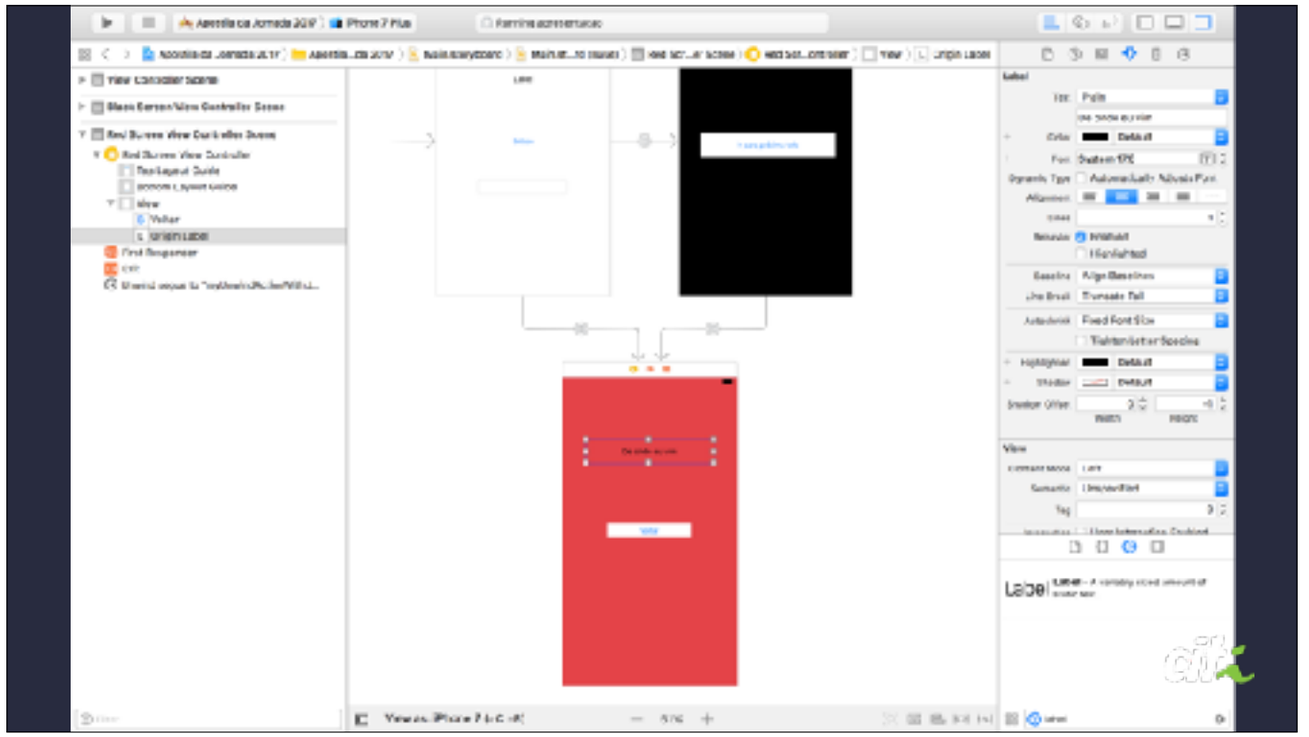
// chamando segue

```swift
// chamando segue


    if tituloLabel.text == "Black" {
        self.performSegue(withIdentifier: "FromVCToBlackScreen", sender: self)

    } else if tituloLabel.text == "Red"{
        self.performSegue(withIdentifier: "FromVCToRedScreen", sender: self)
    }
```

// Passando Dados

// passando dados

- Se precisarmos passar informações entre view controllers
- Podemos passá-las pela segue



// pass

// pass

// passando dados

```
class RedScreenViewController: UIViewController {
    var recievedData: String?

    @IBOutlet weak var originLabel: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()

        self.originLabel.text = self.recievedData

    }
}
```

// passando dados

No ViewController:

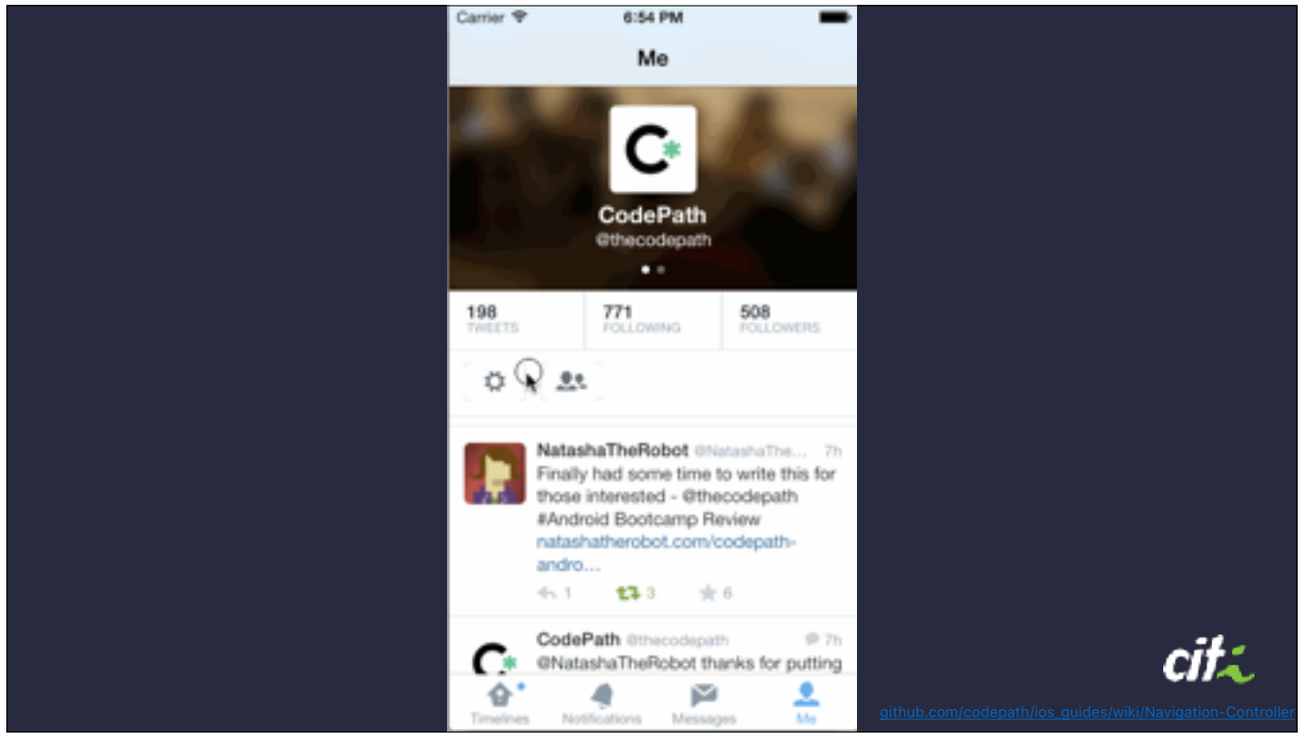
```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if let redVC = segue.destination as? RedScreenViewController {
        redVC.recievedData = "Vim do VC"
    }
}
```

No BlackScreenViewController:

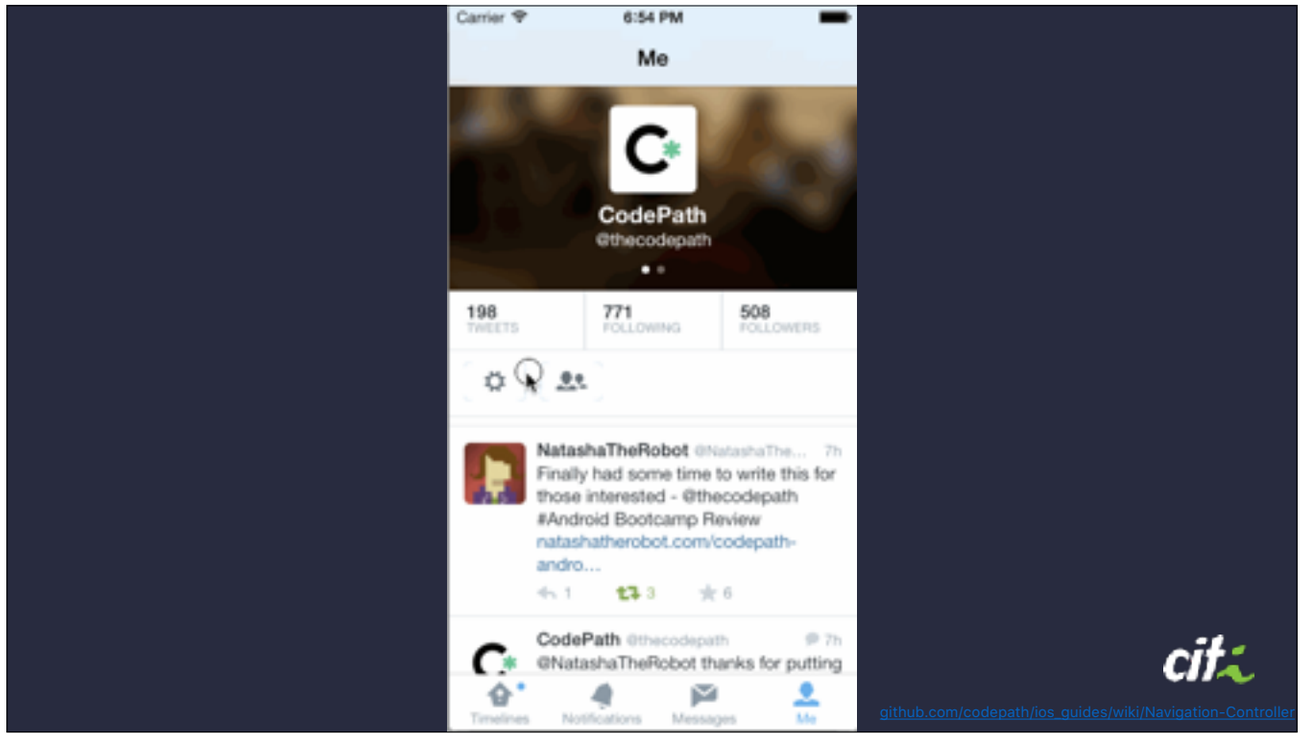
```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if let redVC = segue.destination as? RedScreenViewController {
        redVC.recievedData = "Vim do BlackVC"
    }
}
```

// Navigation Controller

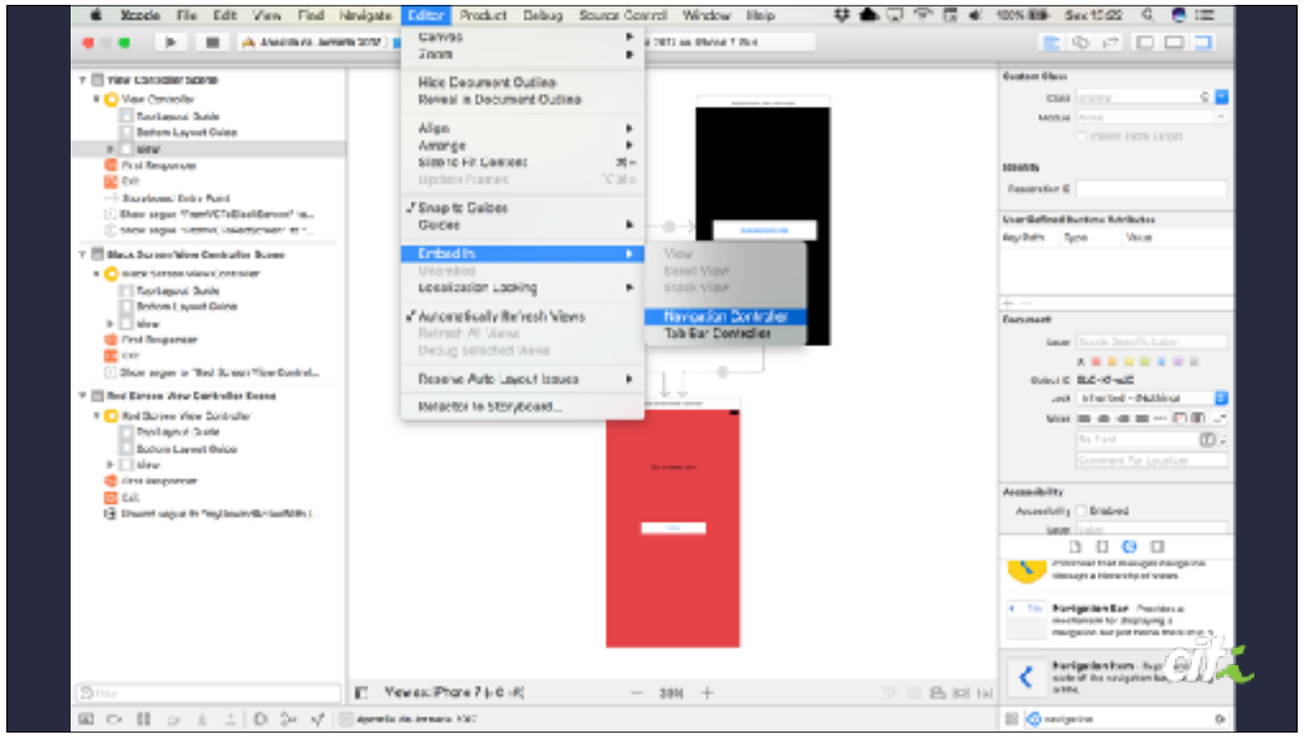
citi

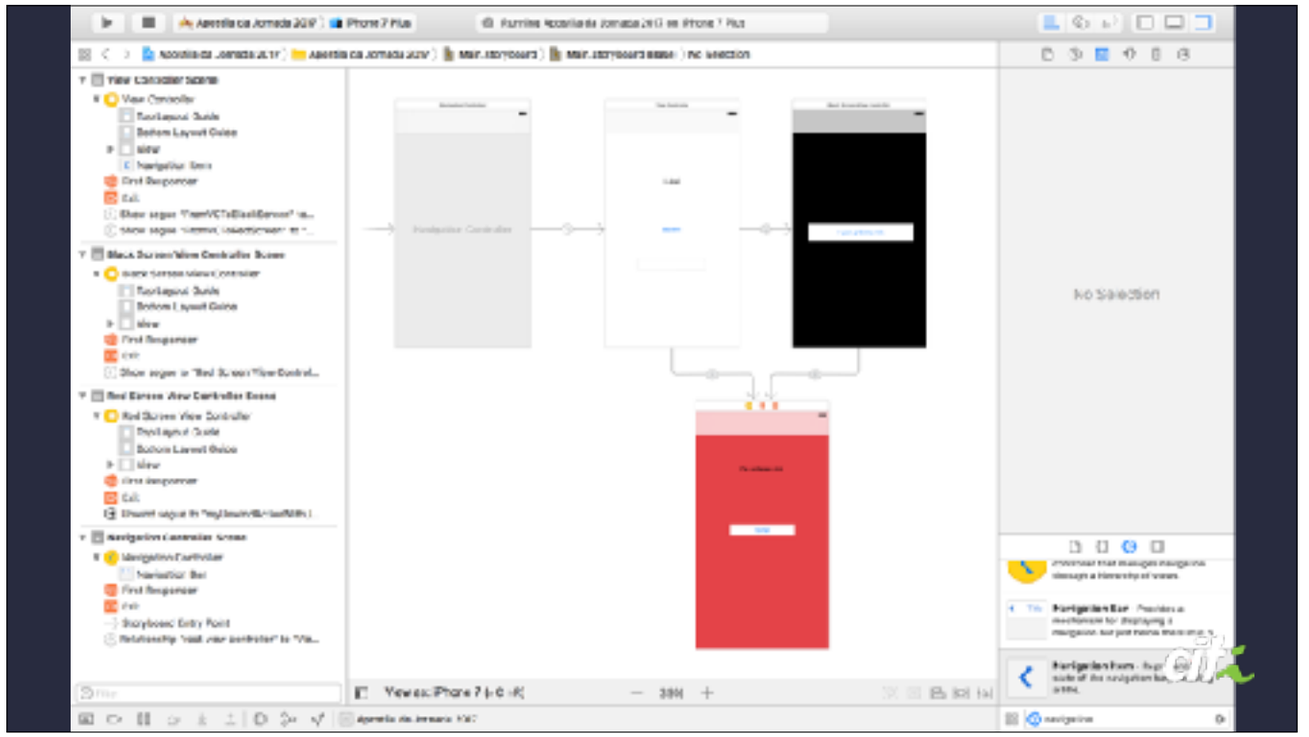


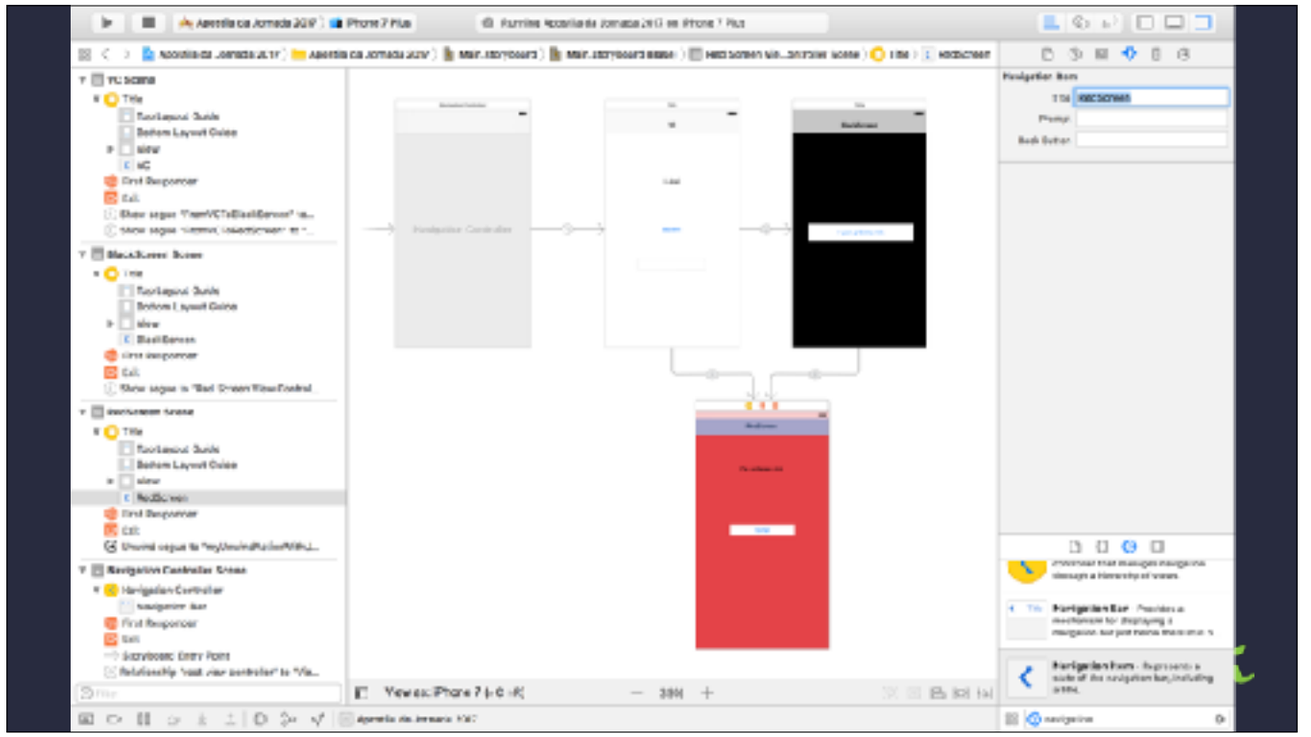
github.com/codepath/ios_guides/wiki/Navigation-Controller



github.com/codepath/ios_guides/wiki/Navigation-Controller


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Hide Document Outline
Reveal in Document Outline
Align
Arrange
Snap to Guides
Guides
Embed In
Localization Testing
Automatically Refresh Views
Refresh All Views
Debug Selected Views
Resolve Auto Layout Issues
Reflect to Storyboard...
View
Navigation Controller
Tab Bar Controller


No Selection



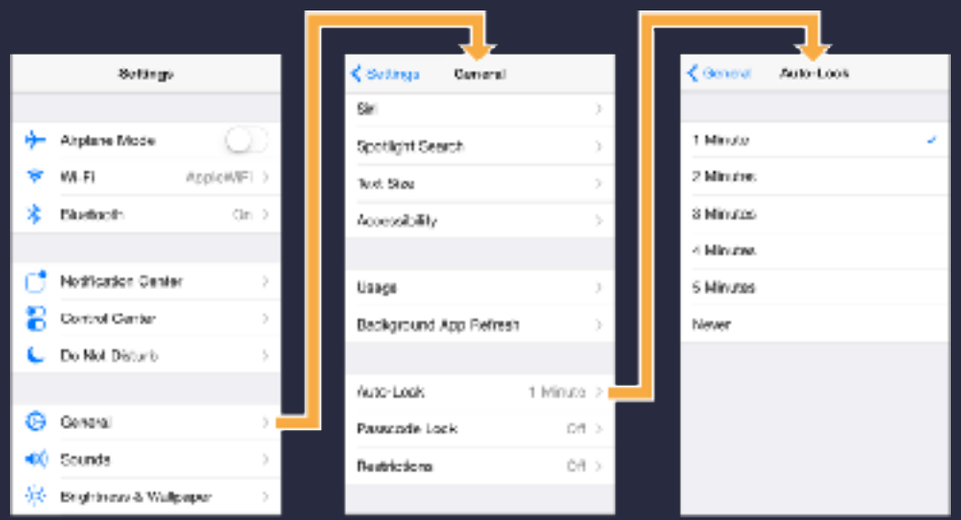
// Navigation controller

- UINavigationController
- Navegação hierárquica
- Pilha de VCs

// Navigation controller

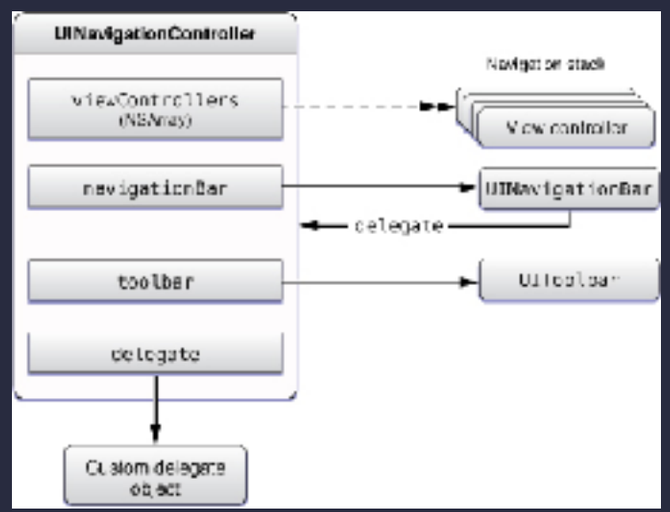
- UINavigationController
- Navegação hierárquica
- Pilha de VCs

// Navigation controller



https://developer.apple.com/documentation/uikit/uinavigationcontroller

// Navigation controller

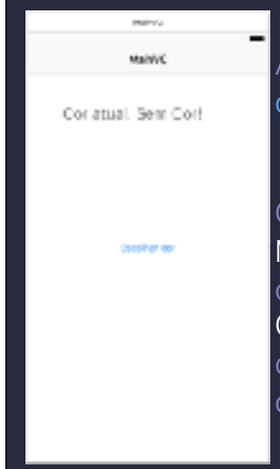


https://developer.apple.com/documentation/uikit/uinavigationcontroller

// Exercício

// Exercício 11-a

Navegação



Apertar em Escolher cor leva para ColorVC

O background de MainVC deve ser da cor escolhida, em ColorVC, e a label deve indicar o nome da cor.

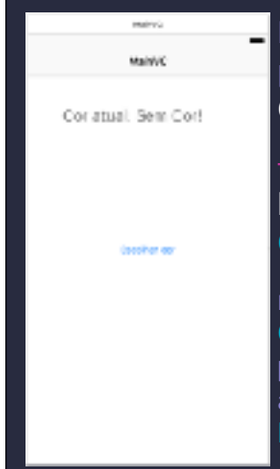
Apertar em um botão de cor muda o conteúdo da **label**.

Apertar em save retorna para o MainVC, passando a **cor** escolhida

adaptado de: https://goo.gl/Bxaeob

// Exercício 11-b

Navegação com Delegate



Implementa um protocolo ColorVCDelegate:

func chosen(color: UIColor, by colorVC: ColorViewController)

retira o ColorViewController da **pilha** de navegação e atualiza interface do MainViewController

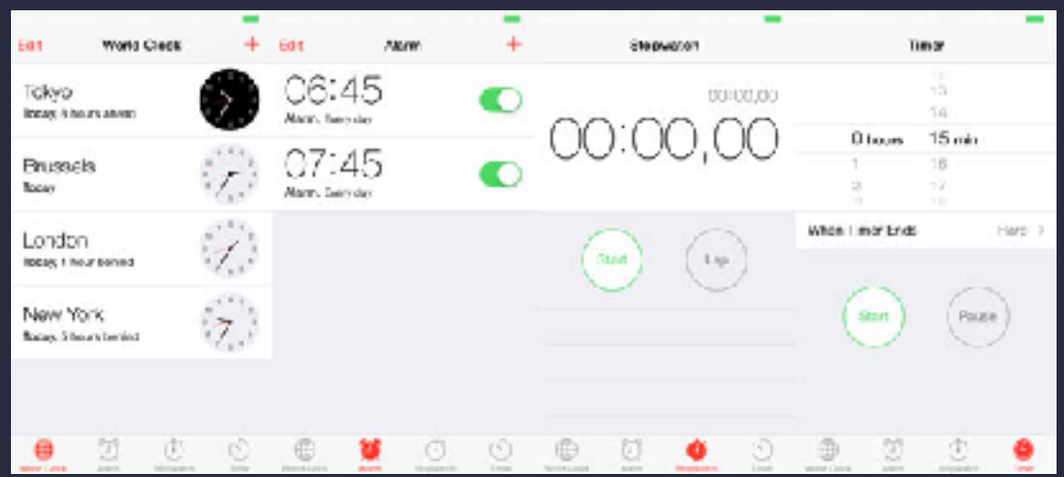
tem uma propriedade **delegate** do tipo ColorVCDelegate

ao apertar save, chama método chosen(color: by:) de seu **delegate**

adaptado de: https://goo.gl/Bxaeob

// Tab Bar

// tab bar



https://goo.gl/U9jLvQ

// tab bar

Gerencia navegação

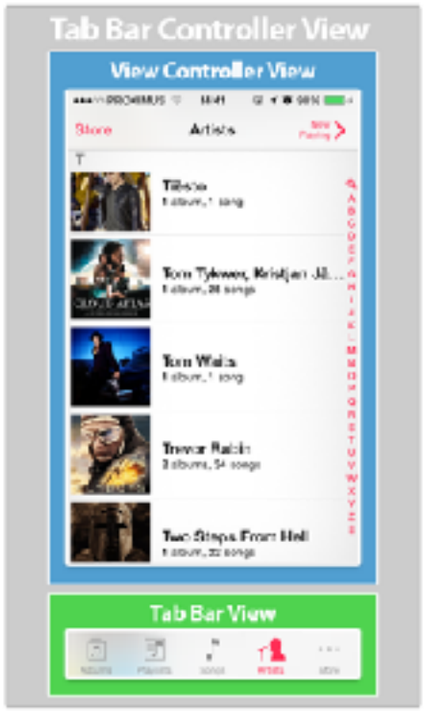
View Controllers não relacionados

Tab bar vs Nav bar

https://goo.gl/U9jLvQ

UITabBarController is another UIViewController subclass.
While navigation controllers manage a stack of related view controllers,
 tab bar controllers manage an array of view controllers that have no explicit relation to one another.

// tab bar

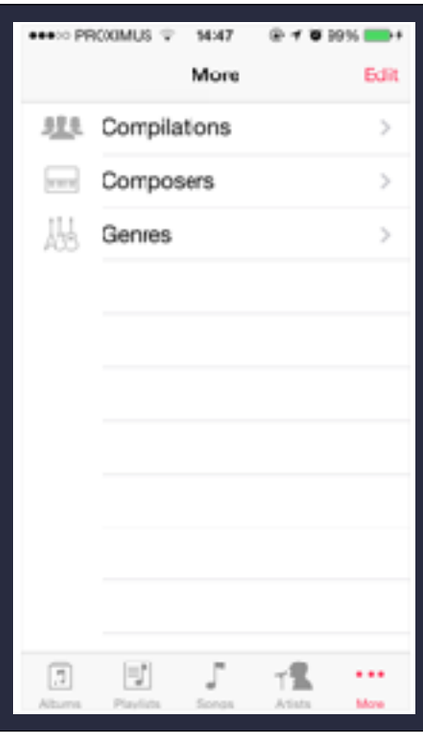


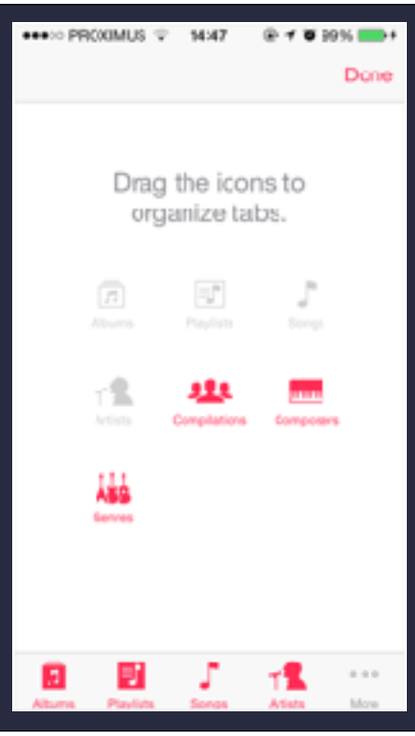
https://goo.gl/U9jLvQ

toda subclasse de UIViewController gerencia views
Tab bar controller gerencia duas:

- A da própria tab bar
- a view do viewController atual

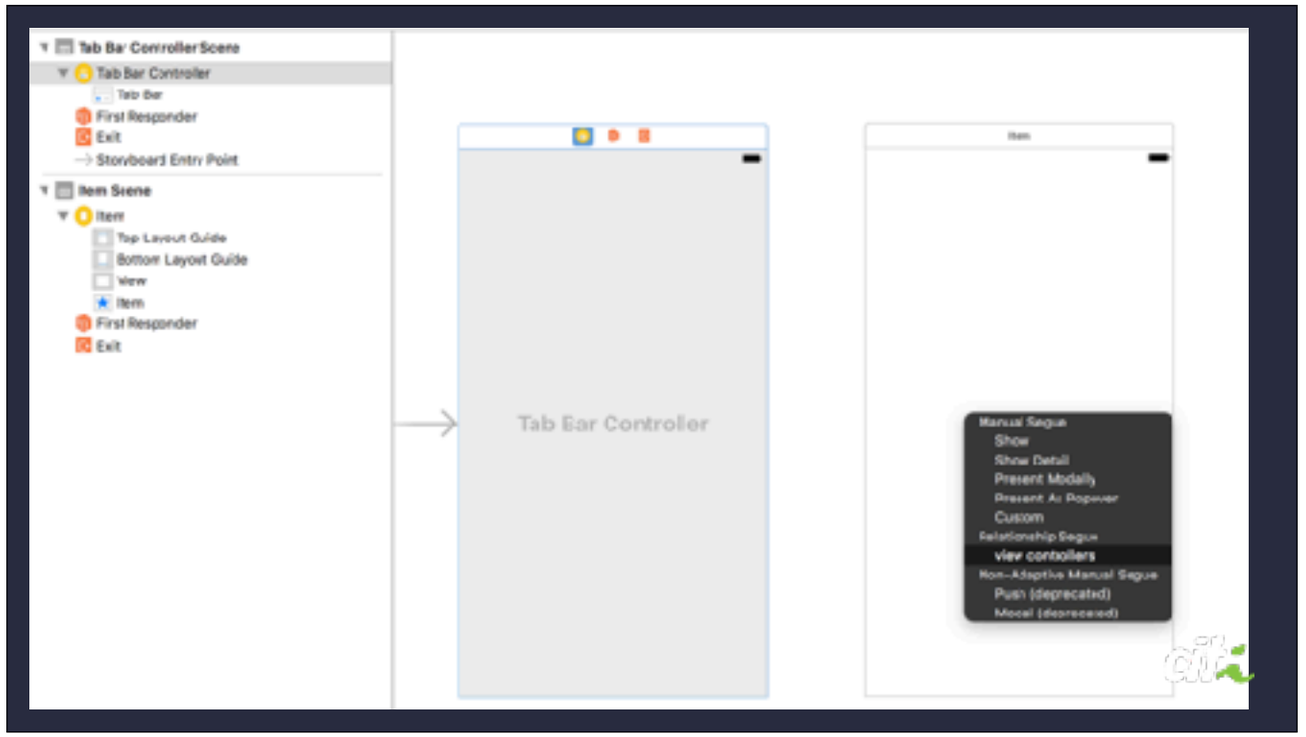
// tab bar



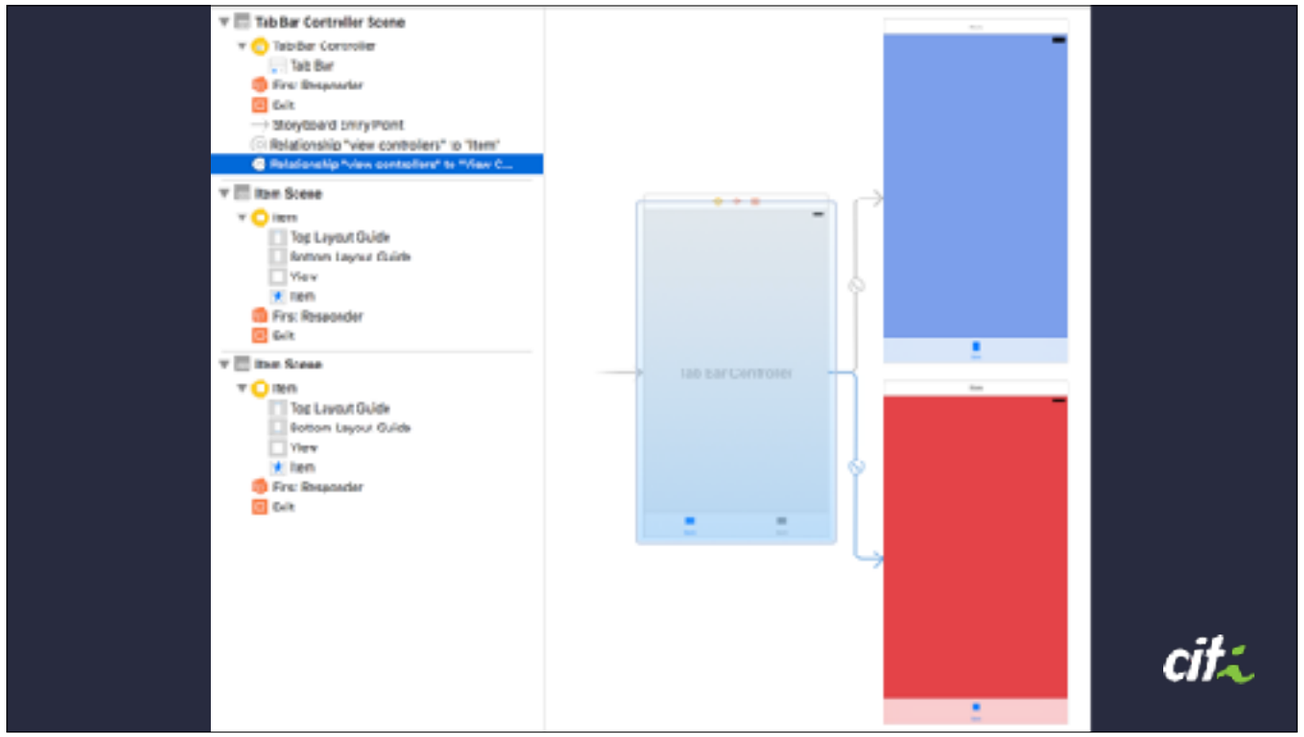


https://goo.gl/U9jLvQ

tab bar view:
- só mostra até 5 itens
- pode gerenciar mais de 5 VCs



relationship segue



adicionando segundo VC

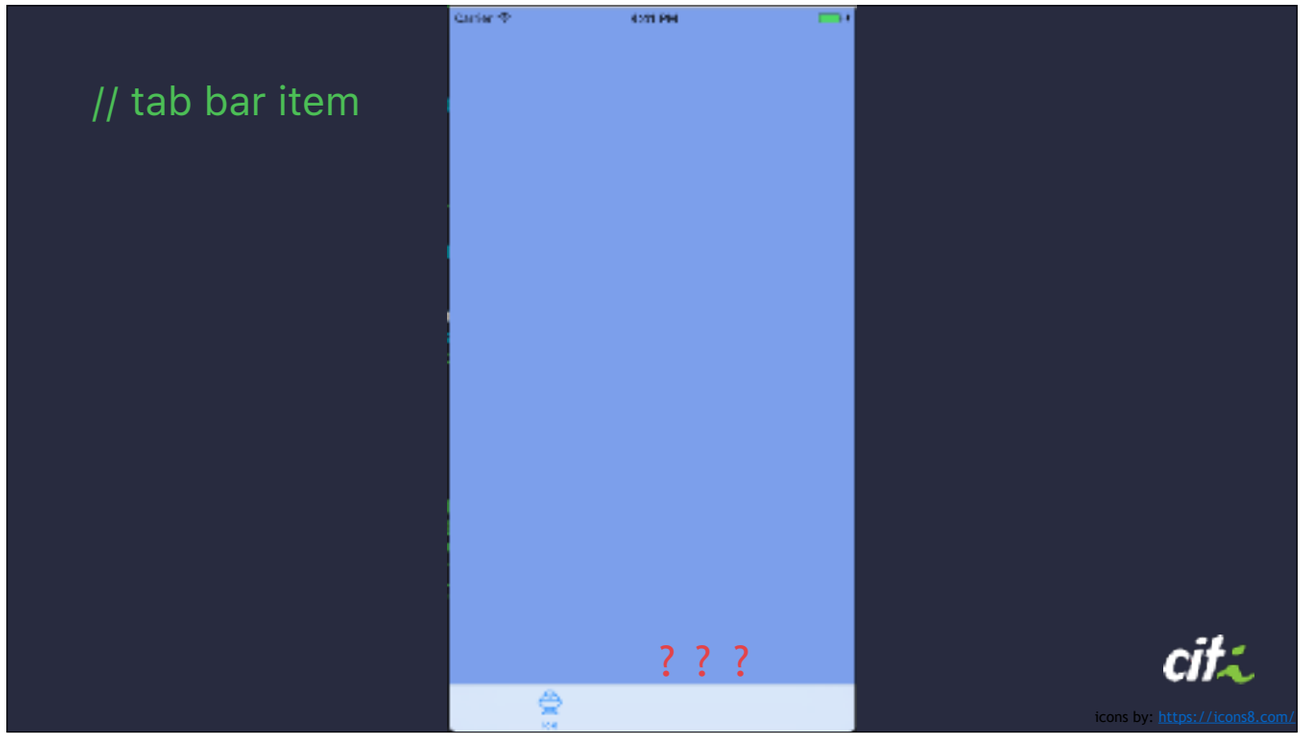
// tab bar item

```swift
class RedViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        // Do any additional setup after loading the view.

        self.tabBarItem = UITabBarItem(title: "Fire", image: fire, tag: 2)
    }
```

```swift
class BlueViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        // Do any additional setup after loading the view.

        self.tabBarItem.title = "Ice"
        self.tabBarItem.image = ice
    }
```

icons by: https://icons8.com/

adicionando item



adicionando item

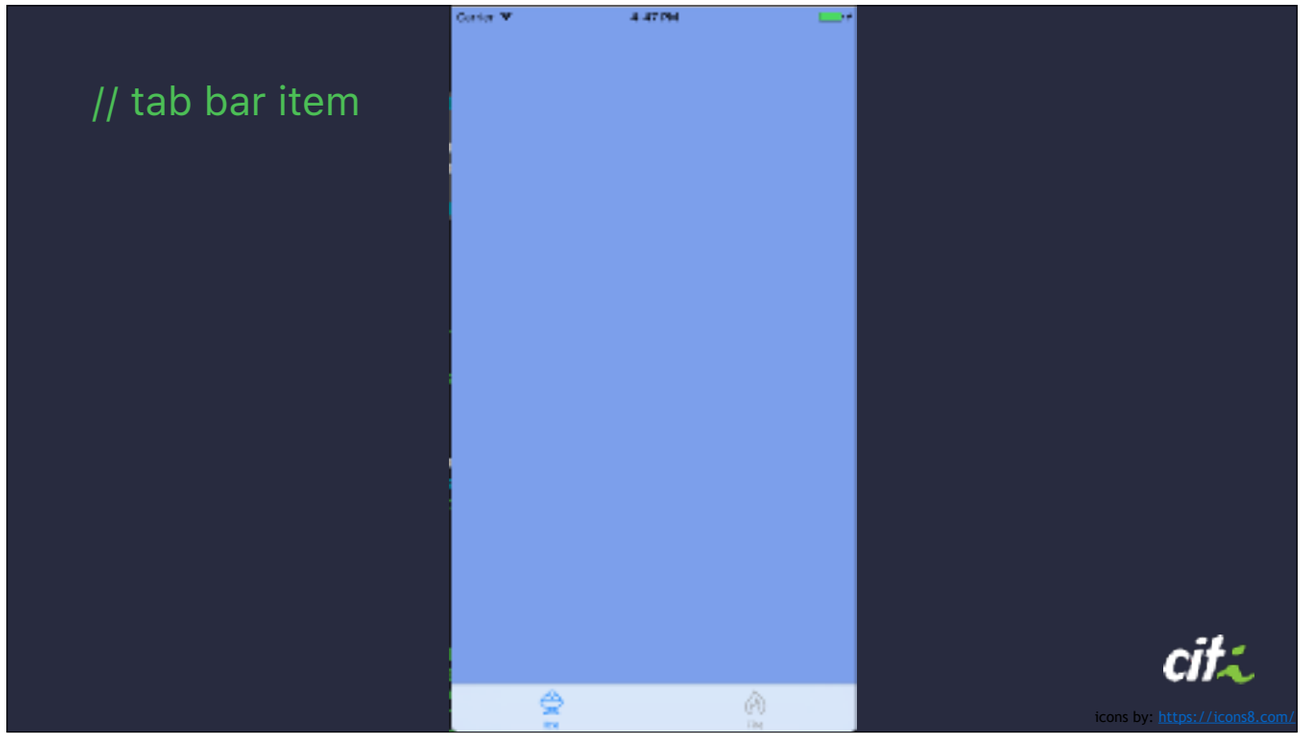
// tab bar item

icons by: https://icons8.com/

adicionando item

```
// tab bar item


    class RedViewController: UIViewController {

        required init?(coder aDecoder: NSCoder) {
            super.init(coder: aDecoder)

            self.tabBarItem = UITabBarItem(title: “Fire",
                                           image: #imageLiteral(resourceName: “fire"),
                                           tag: 2)
        }
```

adicionando item

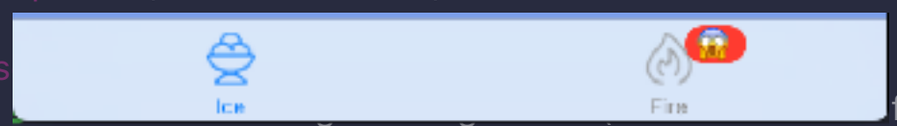


adicionando item

```swift
// tab bar badge

class RedViewController: UIViewController {

    required init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)

        self.tabBarItem = UITabBarItem(title: “Fire",
                                       image: #imageLiteral(resourceName: “fire"),
                                       tag: 2)

        self.tabBarItem.badgeValue = “😱"
    }
```

adicionando item

```
// tab bar badge

class RedViewController: UIViewController {

    required init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
        s
                                                                  fire"),
                                tag: 2)

        self.tabBarItem.badgeValue = "😱"
    }
```



adicionando item

// Exercício

// Exercício 12

Tab bar grande

Fazer uma aplicação com um **TabBarController**

Colocar os seguintes **ViewControllers**:

1. CarroViewController
2. MotoViewController
3. BarcoViewController
4. NavioViewController
5. TremViewController
6. OnibusViewController

Colocar um ícone e um título no **construtor** de cada VC

Colocar uma **ImageView** em cada ViewController com uma imagem correspondente

Tentar **reajustar** os itens mostrados na tab bar

// Desafio 01

Tab e Nav

Junte os últimos dois exercícios, para que sua aplicação final contenha uma **Tab Bar**, e uma **Nav Bar** ao mesmo tempo.

// Ementa

EMENTA PRELIMINAR DO CURSO:

SWIFT	STORYBOARD
Variáveis, constantes e operadores	Views
Tipos de variáveis (números, strings, entre outras)	InputView
Coleções (array, dicionário)	ScrollView
Controle de fluxo (condicionais e loops)	TableView
Funções e closures	Navigation Controller
Enums	Tab Bar
Classes e structs	Labels
Protocols	Botões
Casting de tipos	Trocando de telas
Optionals	Integrando o storyboard com o código (outlets, buttons, entre outros)
Persistência	Webviews

COMPLEMENTO: XCODE, MONETIZAÇÃO E DESIGN DE INTERFACE.

EMENTA PRELIMINAR DO CURSO:

SWIFT	STORYBOARD
Variáveis, constantes e operadores	Views
Tipos de variáveis (números, strings, entre outros)	InputView TextField
Coleções (array, dicionário)	ScrollView
Controle de fluxo (condicionais e loops)	TableView
Funções e closures	Navigation Controller
Enums	Tab Bar
Classes e structs	Labels
Protocols	Buttons
Casting de tipos	Trocando de telas
Optionals Delegation	Integrando o storyboard com o código (outlets, buttons, entre outros)
Persistência	Web views

COMPLEMENTO: MONETIZAÇÃO E DESIGN DE INTERFACE

// Proposta semana 02

Segunda	Terça	Quarta	Quinta	Sexta
TableView	WebView	Projeto	Projeto	Projeto
ScrollView	Persistência		Extra	Extra


DÚVIDAS
?
citi