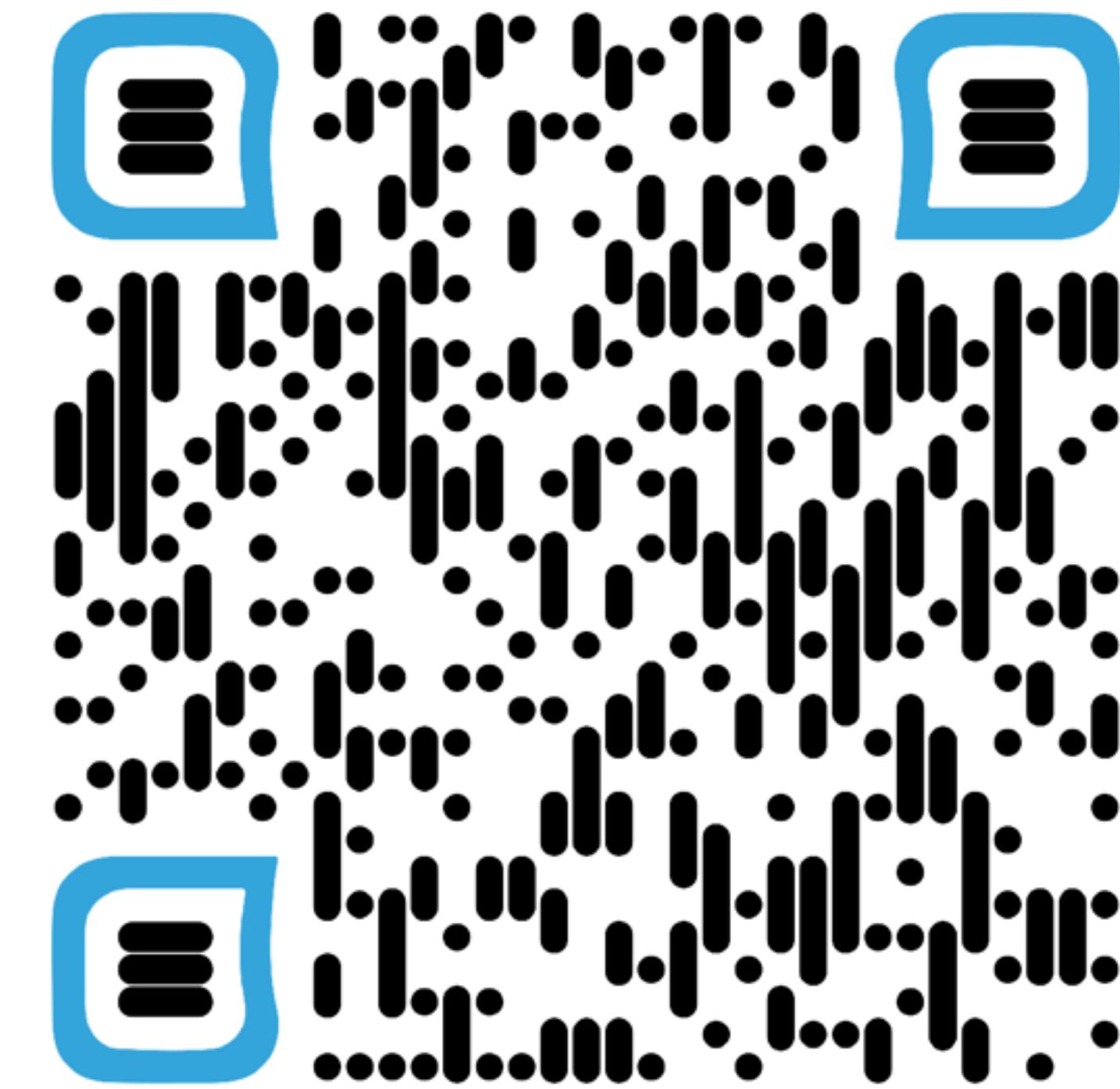




PROF. PEDRO HENRIQUE

DESENVOLVIMENTO PARA IOS 11 COM SWIFT 4

GITHUB DA TURMA



[HTTPS://GIT.IO/VF50W](https://git.io/vf50w)

ONDE ENCONTRAR O MATERIAL?

LEIA O QR CODE

ALÉM DO TRADICIONAL BLACKBOARD DO IESB

AGENDA

- ▶ Emoticon: aplicando o MVC
- ▶ Gestos
- ▶ Emoticon: modificando a expressão facial com gestos
- ▶ Múltiplos MVCS
 - ▶ Tab Bar, Navegação, Split View Controller





ESTÁ SE
CONCENTRANDO NO
PROBLEMA. ASSIM,
NÃO PODE VER A
SOLUÇÃO.

Patch Adams

GESTOS

- ▶ Já vimos como desenhar em uma UIView. Como vamos detectar toques agora? Existe a possibilidade de sermos notificados dos dados brutos relacionados aos eventos de toque. Ou, podemos reagir a certos gestos pré-definidos.
- ▶ Todos os gestos são reconhecidos por instâncias de UIGestureRecognizer. A classe base é “abstrata”. Podemos apenas usar as filhas concretas.
- ▶ Existem duas formas de usar um UIGestureRecognizer:
 1. Adicionar uma instância à UIView (a view reconhece aquele gesto)
 2. Implementar um método para “reconhecer” o gesto (não necessariamente tratado pela view)

GESTOS

- ▶ Normalmente, o primeiro caso é realizado através de um Controller, apesar de ocasionalmente a UIView ser capaz de fazer o reconhecimento pleno dos gestos
- ▶ No segundo caso, o “reconhecedor” dos gestos pode ser tanto a UIView quanto o Controller
- ▶ Qual é o certo? Depende da situação!
- ▶ Vamos ver exemplos de ambos daqui a pouco!

ADICIONANDO UM GESTO

- ▶ Imagine que você quer que uma UIView reconheça o gesto de pinça (Pan)
- ▶ Podemos fazer isso de forma muito simples com um observador no Outlet daquela UIView...

```
@IBOutlet weak var pannableView: UIView {
    didSet {
        let action = #selector(ViewController.pan(recognizer:))
        let panGestureRecognizer = UIPanGestureRecognizer(target: self, action: action)
        pannableView.addGestureRecognizer(panGestureRecognizer)
    }
}
```

ADICIONANDO UM GESTO

- ▶ Imagine que você quer que uma UIView reconheça o gesto de pinça (Pan)
- ▶ Podemos fazer isso de forma muito simples com um observador no Outlet daquela UIView...

```
@IBOutlet weak var pannableView: UIView {  
    didSet {  
        let action = #selector(ViewController.pan(recognizer:))  
        let panGestureRecognizer = UIPanGestureRecognizer(target: self, action: action)  
        pannableView.addGestureRecognizer(panGestureRecognizer)  
    }  
}
```

O observador (`didSet`) é chamado quando o iOS liga o Outlet em tempo de execução.

ADICIONANDO UM GESTO

- ▶ Imagine que você quer que uma UIView reconheça o gesto de pinça (Pan)
- ▶ Podemos fazer isso de forma muito simples com um observador no Outlet daquela UIView...

```
@IBOutlet weak var pannableView: UIView {
    didSet {
        let action = #selector(ViewController.pan(recognizer:))
        let panGestureRecognizer = UIPanGestureRecognizer(target: self, action: action)
        pannableView.addGestureRecognizer(panGestureRecognizer)
    }
}
```

Aqui estamos criando uma instância de uma subclasse de UIGestureRecognizer (especializada no gesto de pinça)

ADICIONANDO UM GESTO

- ▶ Imagine que você quer que uma UIView reconheça o gesto de pinça (Pan)
- ▶ Podemos fazer isso de forma muito simples com um observador no Outlet daquela UIView...

```
@IBOutlet weak var pannableView: UIView {
    didSet {
        let action = #selector(ViewController.pan(recognizer:))
        let panGestureRecognizer = UIPanGestureRecognizer(target: self, action: action)
        pannableView.addGestureRecognizer(panGestureRecognizer)
    }
}
```

O target é notificado quando o gesto é reconhecido. Neste caso, é o próprio ViewController.

ADICIONANDO UM GESTO

- ▶ Imagine que você quer que uma UIView reconheça o gesto de pinça (Pan)
- ▶ Podemos fazer isso de forma muito simples com um observador no Outlet daquela UIView...

```
@IBOutlet weak var pannableView: UIView {
    didSet {
        let action = #selector(ViewController.pan(recognizer:))
        let panGestureRecognizer = UIPanGestureRecognizer(target: self, action: action)
        pannableView.addGestureRecognizer(panGestureRecognizer)
    }
}
```

A action representa o método que será evocado quando o gesto for reconhecido. O método recebe um próprio **panGestureRecognizer**.

ADICIONANDO UM GESTO

- ▶ Imagine que você quer que uma UIView reconheça o gesto de pinça (Pan)
- ▶ Podemos fazer isso de forma muito simples com um observador no Outlet daquela UIView...

```
@IBOutlet weak var pannableView: UIView {
    didSet {
        let action = #selector(ViewController.pan(recognizer:))
        let panGestureRecognizer = UIPanGestureRecognizer(target: self, action: action)
        pannableView.addGestureRecognizer(panGestureRecognizer)
    }
}
```

Aqui é onde solicitamos que a View passe a reconhecer ao gesto dentro da área bounds. Agora vamos falar sobre como implementar o método que lida com o reconhecimento...

RECONHECENDO UM GESTO

- ▶ Um método que reconhece gestos precisa de informações específicas acerca do gesto. Então, cada classe concreta filha de `UIGestureRecognizer` provê meios de obter informações relevantes sobre o respectivo tipo de gesto;
- ▶ O **UIPanGestureRecognizer**, por exemplo, fornece três métodos:
`func translation(in: UIView?) -> CGPoint` //chamada cumulativa desde o início
`func velocity(in: UIView?) -> CGPoint` //quão rápido os dedos se movem (pt/s)
`func setTranslation(CGPoint, in: UIView?)` //este é interessante porque te permite resetar a translação. Ao fazer isto, você acaba fazendo com que a translação seja "incremental".

RECONHECENDO UM GESTO

- ▶ A superclasse também oferece a informação sobre o estado do gesto:
var state: UIGestureRecognizerState { get }
- ▶ O valor da propriedade permanece como **.possible** até que o gesto seja reconhecido
- ▶ Em gestos continuados (como o de pinça), o valor passa para **.began** e depois fica repetidamente em **.changed** até que vai para **.ended** quando o gesto acaba
- ▶ Em gestos discretos (como um swipe), o valor vai direto para **.ended** ou **.recognized**
- ▶ Em certas ocasiões, o valor pode passar a ser **.failed** ou **.cancelled**, então vamos ficar de olho nestes também.

SOU UM INTÉPRETE!!!

RECONHECENDO...

- ▶ Então, dadas estas informações, como um método que trata um gesto de pinça se parece?

```
func pan(recognizer: UIPanGestureRecognizer) {  
    switch recognizer.state {  
    case .changed: fallthrough  
    case .ended:  
        let translation = recognizer.translation(in: pannableView)  
        print(translation) //só para tirar o warning e ficar bonito no slide  
        // atualizar tudo que dependa do gesto de pinça usando translation.x e .y  
        recognizer.setTranslation(CGPoint.zero, in: pannableView)  
    default: break  
    }  
}
```

Lembre-se que a action foi

```
let action = #selector(ViewController.pan(recognizer:))
```

SOU UM INTÉPRETE!!!

RECONHECENDO...

- ▶ Então, dadas estas informações, como um método que trata um gesto de pinça se parece?

```
func pan(recognizer: UIPanGestureRecognizer) {  
    switch recognizer.state {  
    case .changed: fallthrough  
    case .ended:  
        let translation = recognizer.translation(in: pannableView)  
        print(translation) //só para tirar o warning e ficar bonito no slide  
        // atualizar tudo que dependa do gesto de pinça usando translation.x e .y  
        recognizer.setTranslation(CGPoint.zero, in: pannableView)  
    default: break  
    }  
}
```

NÓS SÓ VAMOS FAZER ALGUMA COISA QUANDO OS DEDOS SE MOVEREM OU DEIXAREM A TELA DO DISPOSITIVO.

RECONHECENDO...

- ▶ Então, dadas estas informações, como um método que trata um gesto de pinça se parece?

```
func pan(recognizer: UIPanGestureRecognizer) {  
    switch recognizer.state {  
        case .changed: fallthrough  
        case .ended:  
            let translation = recognizer.translation(in: pannableView)  
            print(translation) //só para tirar o warning e ficar bonito no slide  
            // atualizar tudo que dependa do gesto de pinça usando translation.x e .y  
            recognizer.setTranslation(CGPoint.zero, in: pannableView)  
        default: break  
    }  
}
```

fallthrough significa “execute o código do próximo case”

SOU UM INTÉPRETE!!!

RECONHECENDO...

- ▶ Então, dadas estas informações, como um método que trata um gesto de pinça se parece?

```
func pan(recognizer: UIPanGestureRecognizer) {  
    switch recognizer.state {  
        case .changed: fallthrough  
        case .ended:  
            let translation = recognizer.translation(in: pannableView)  
            print(translation) //só para tirar o warning e ficar bonito no slide  
            // atualizar tudo que dependa do gesto de pinça usando translation.x e .y  
            recognizer.setTranslation(CGPoint.zero, in: pannableView)  
        default: break  
    }  
}
```

AQUI É ONDE PEGAMOS A LOCALIZAÇÃO DO GESTO DENTRO DO SISTEMA DE COORDENADAS DA VIEW.

AGORA PODEMOS FAZER O QUEM ENTENDERMOS COM A INFORMAÇÃO

SOU UM INTÉPRETE!!!

RECONHECENDO...

- ▶ Então, dadas estas informações, como um método que trata um gesto de pinça se parece?

```
func pan(recognizer: UIPanGestureRecognizer) {  
    switch recognizer.state {  
        case .changed: fallthrough  
        case .ended:  
            let translation = recognizer.translation(in: pannableView)  
            print(translation) //só para tirar o warning e ficar bonito no slide  
            // atualizar tudo que dependa do gesto de pinça usando translation.x e .y  
            recognizer.setTranslation(CGPoint.zero, in: pannableView)  
        default: break  
    }  
}
```

AO “RESETAR” A TRANSLATION, FAREMOS COM
QUE NA PRÓXIMA ENTRADA NESTE MÉTODO O
VALOR SEJA INCREMENTAL.

GESTOS

► **UIPinchGestureRecognizer**

`var scale: CGFloat // pode ser alterado (pode "resetar")`
`var velocity: CGFloat { get } // fator de escala por segundo`

► **UIRotationGestureRecognizer**

`var rotation: CGFloat // pode ser alterado (pode "resetar"); valor em radianos`
`var velocity: CGFloat { get } // radianos por segundo`

► **UISwipeGestureRecognizer**

Permite configurar a direção e o número de dedos envolvidos
`var direction: UISwipeGestureRecognizerDirection // direção`
`var numberOfTouchesRequired: Int // número de dedos`

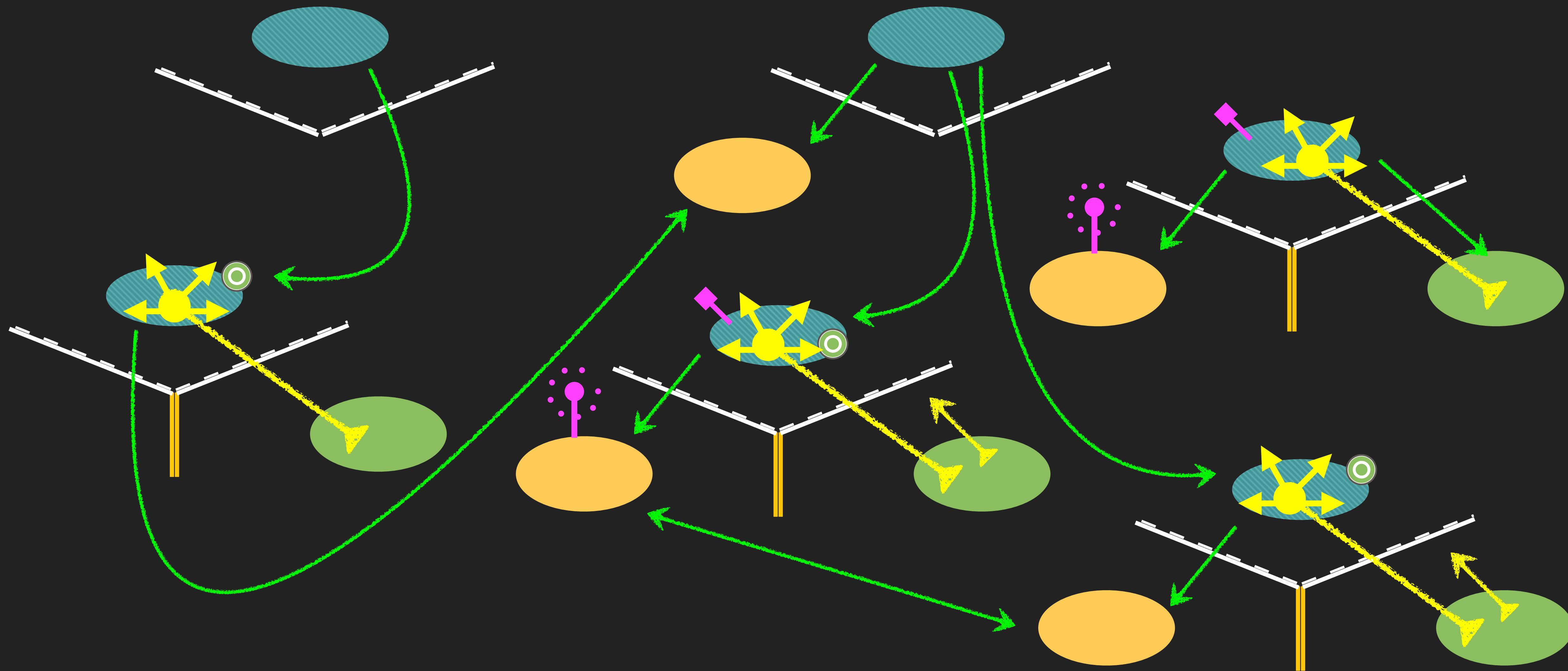
► **UITapGestureRecognizer**

Permite configurar quantos toques e o número de dedos
`var numberOfTapsRequired: Int // um toque, toque duplo, toque triplo, etc...`
`var numberOfTouchesRequired: Int // quantidade de dedos`

RECONHECIMENTO DE GESTOS

HORA DE PRATICAR

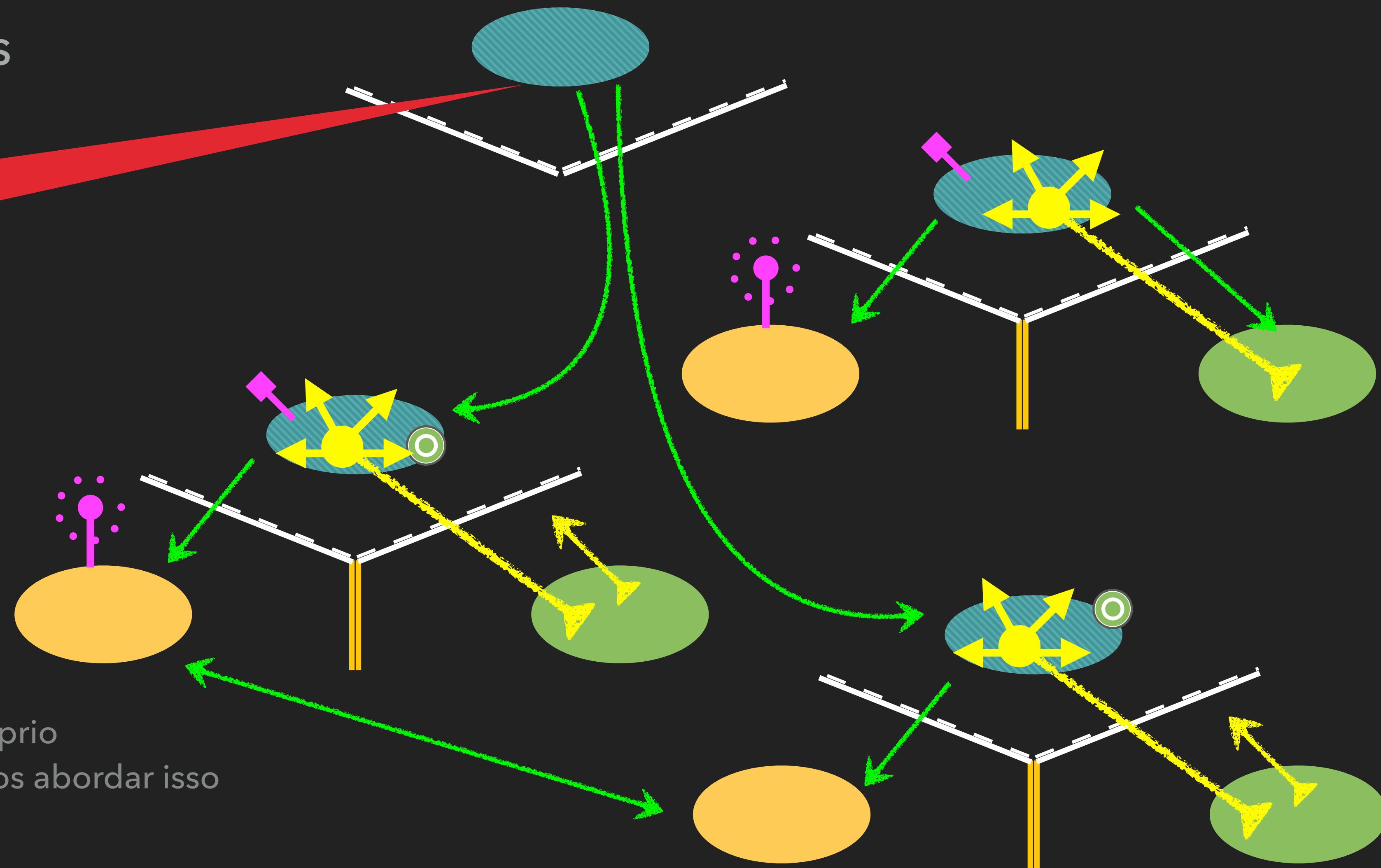
MVCS TRABALHANDO JUNTOS



HORA DE FAZER APLICATIVOS SUPER PODEROSOS!!!

- Para fazer isso, precisamos combinar vários MVCS...

O iOS oferece alguns ViewControllers cujas views são outros MVCS.



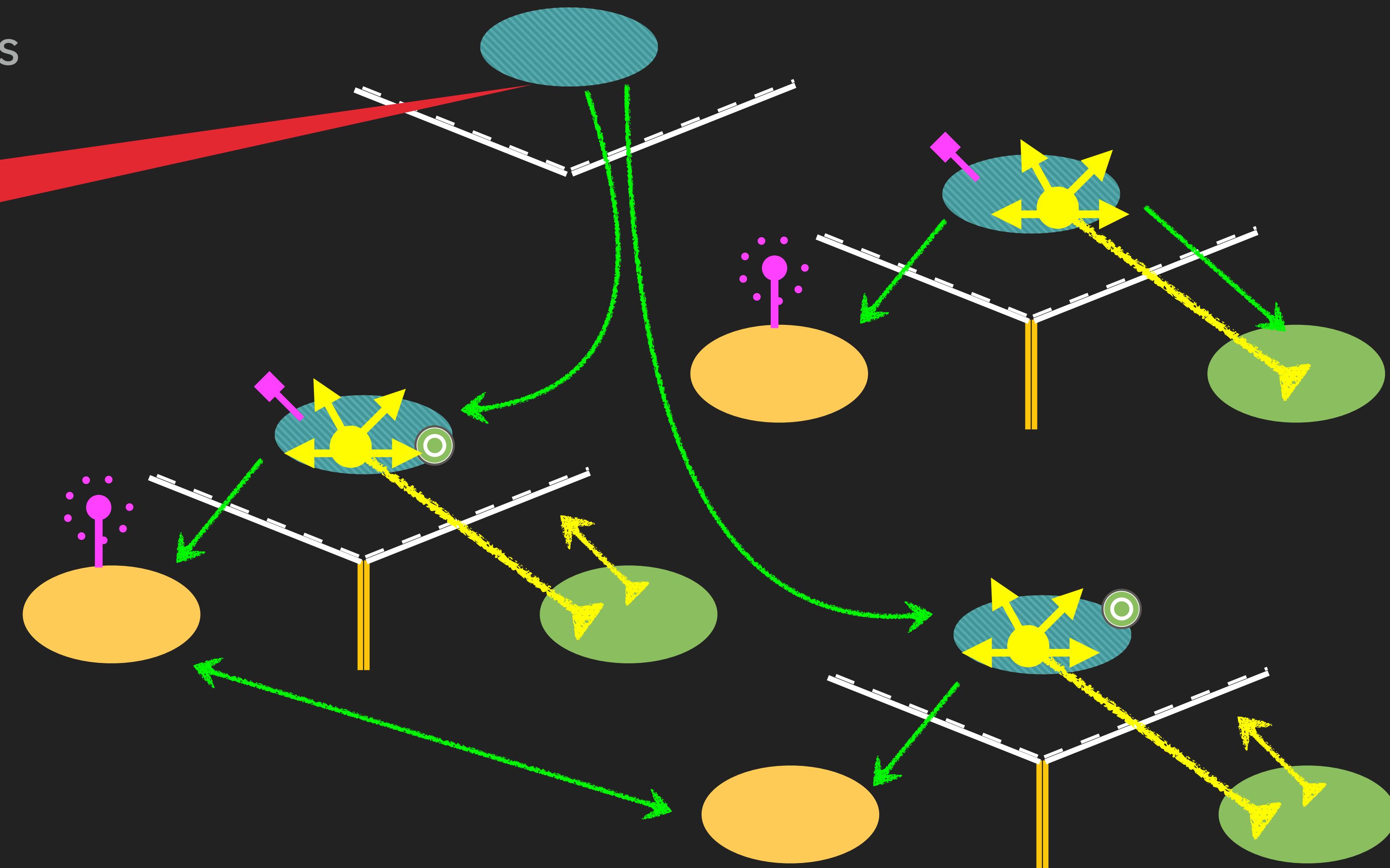
* você mesmo poderia implementar seu próprio Controller capaz de fazer isso, mas não vamos abordar isso neste curso. Por enquanto...

HORA DE FAZER APLICATIVOS SUPER PODEROSOS!!!

- Para fazer isso, precisamos combinar vários MVCS...

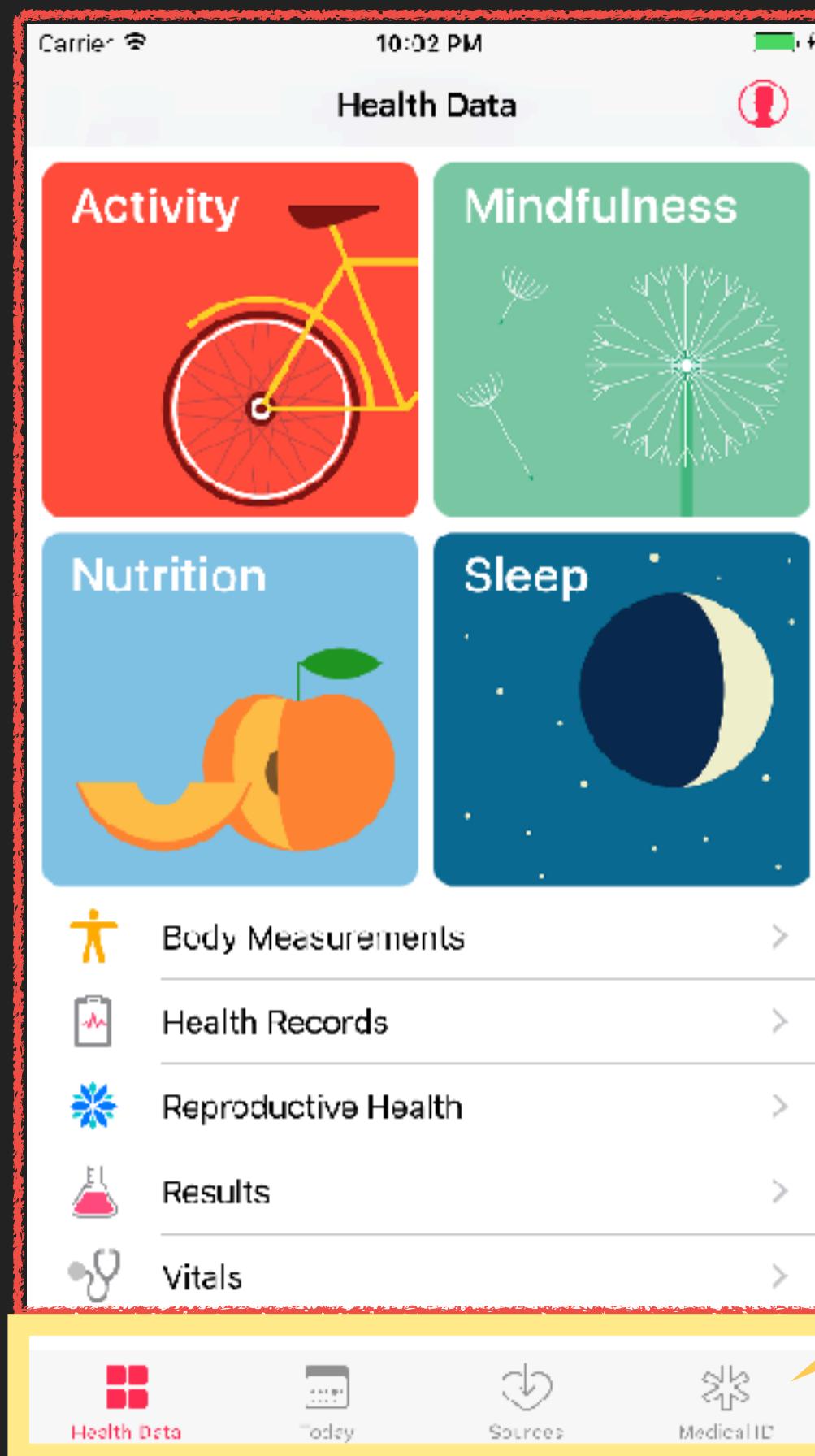
O iOS oferece alguns ViewControllers cujas views são outros MVCS.

UITabBarController
UISplitViewController
UINavigationController



UITABBARCONTROLLER

- Deixa você escolher entre MVCs diferentes...



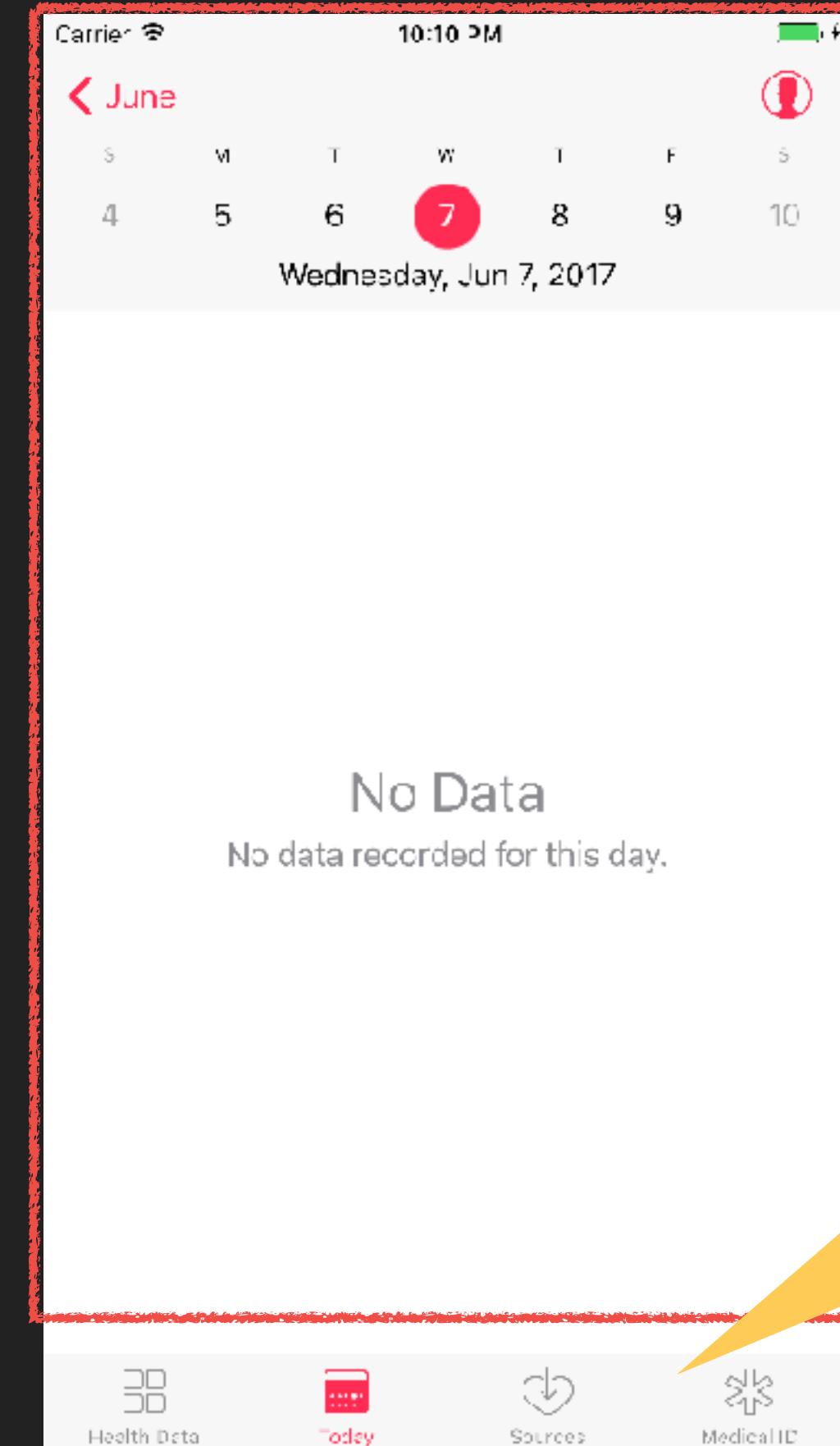
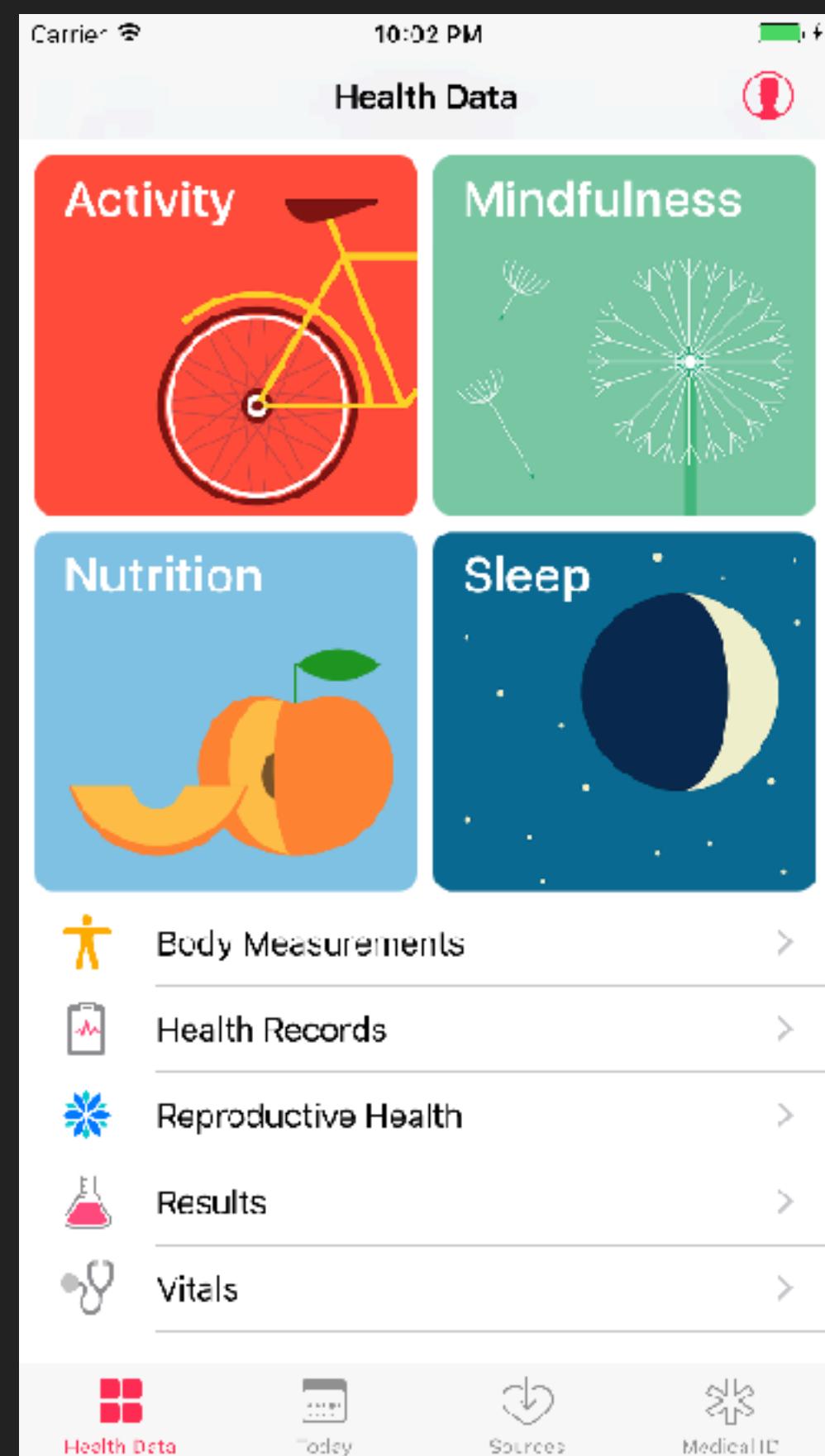
UM MVC CHAMADO “HEALTH DATA”

O ícone e o título (e ainda a pequena badge vermelha que pode aparecer) destas abas são determinados pelos próprios MVCs através da propriedade tabBarItem, que todo View controller tem. Mas o normal é configurar este tipo de coisa através do storyboard.

ABAS!!!

UITABBARCONTROLLER

- Deixa você escolher entre MVCs diferentes...



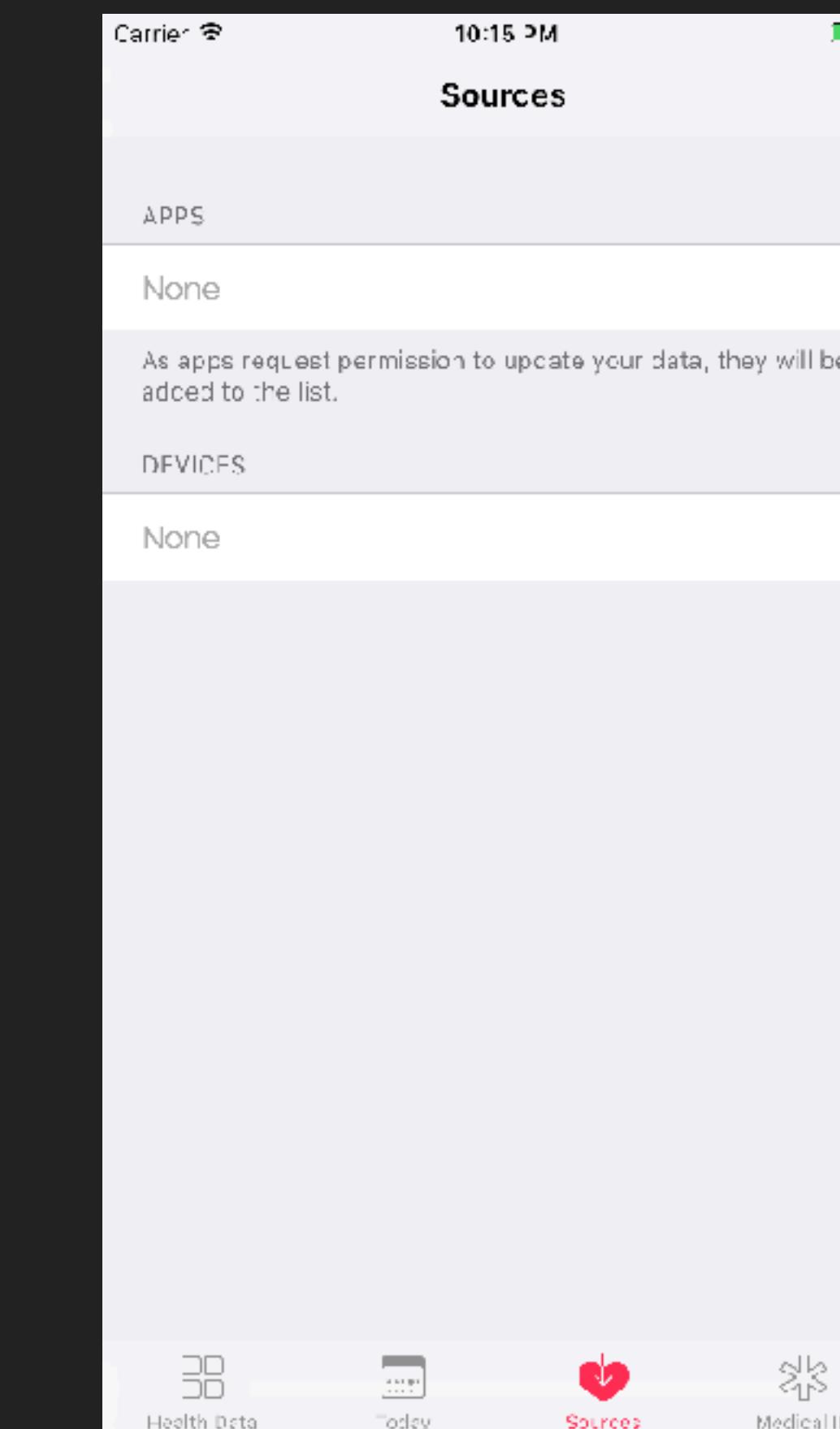
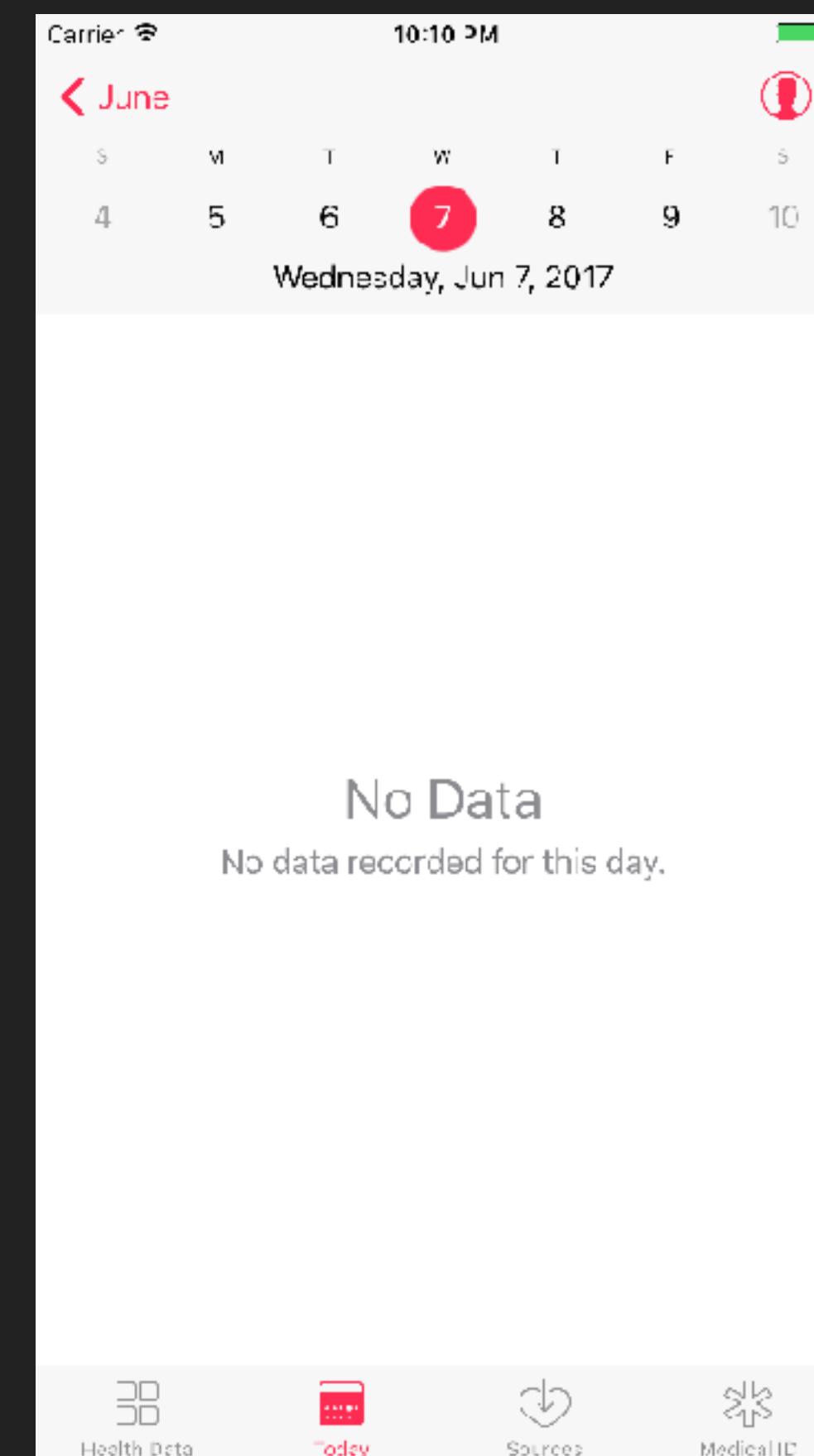
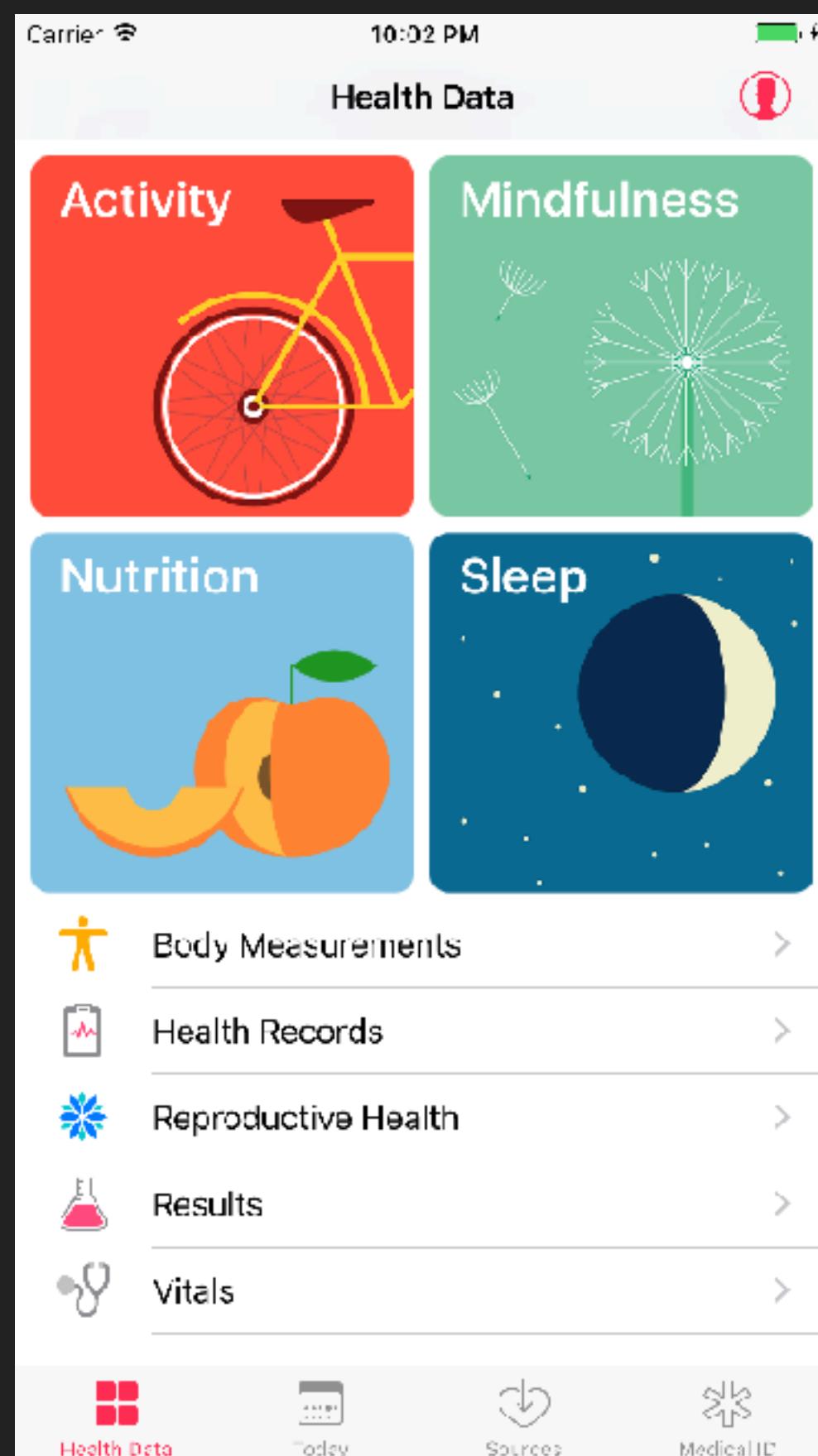
UM MVC CHAMADO "TODAY"

Se houverem muitas abas para colocar aqui, o UITabBarController vai automaticamente oferecer uma interface para o usuário gerenciar o excesso (o usuário poderá escolher o que quer ver e/ou poderá arrastar de/para a barra de abas os MVCs que desejar)

ABAS!!!

UITABBARCONTROLLER

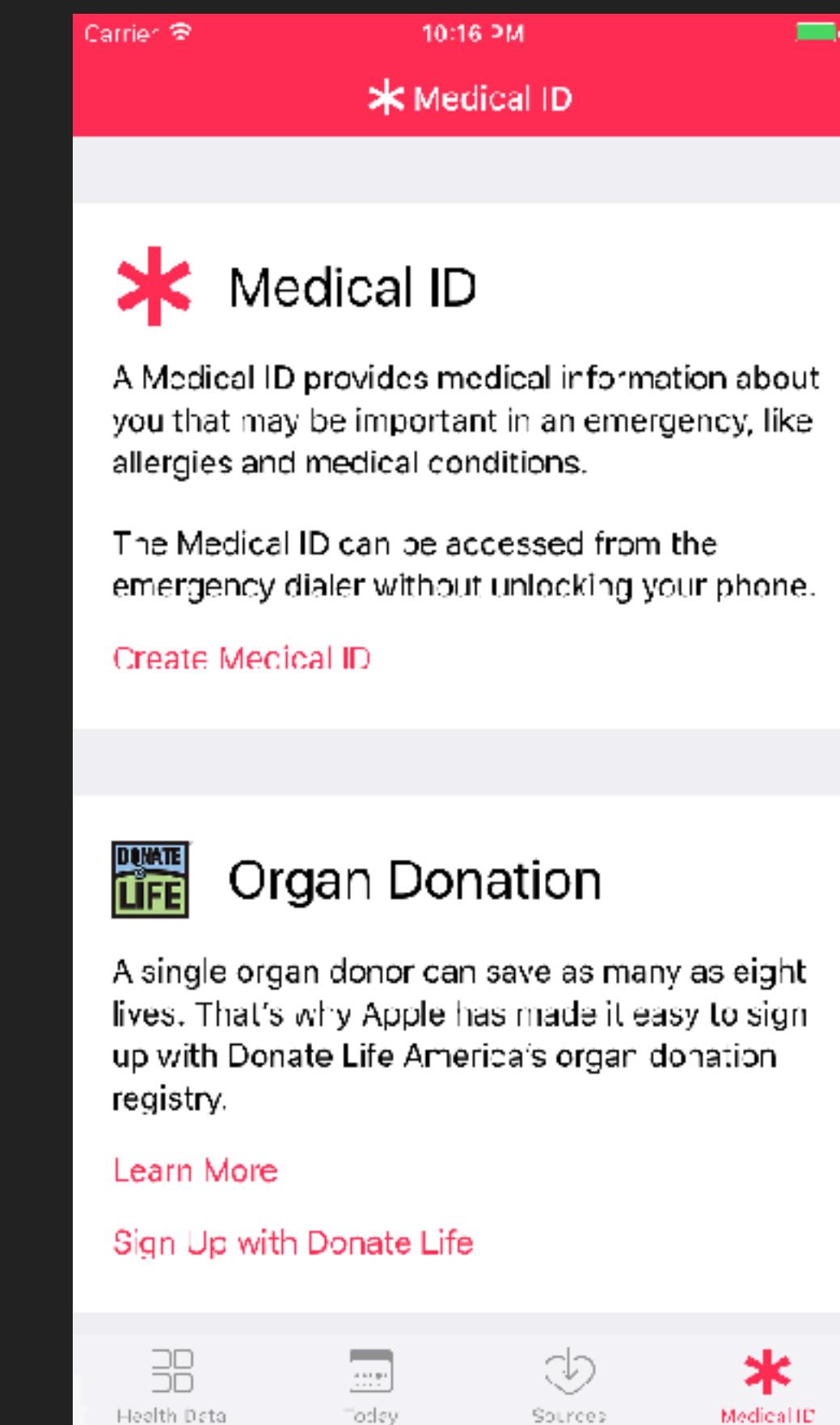
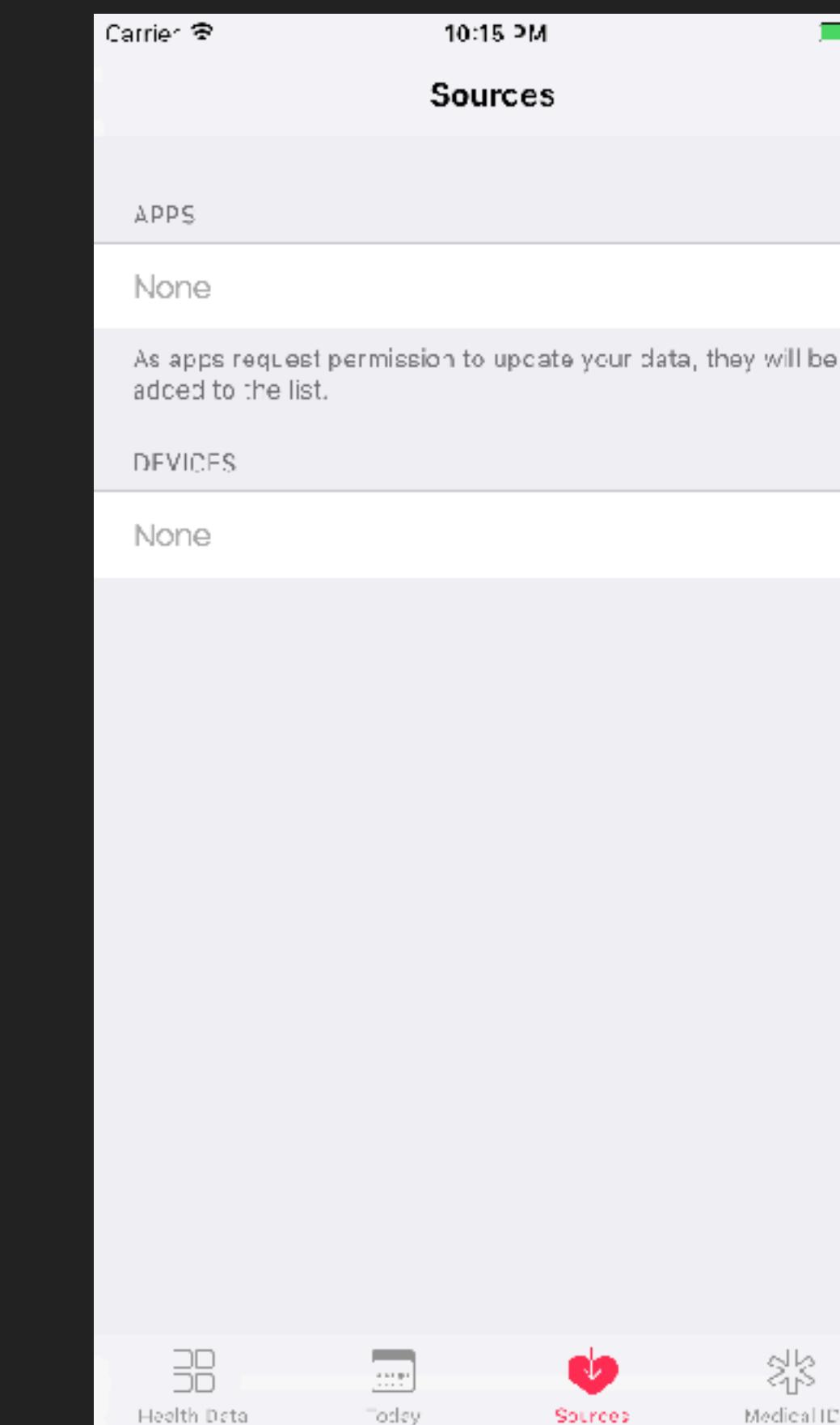
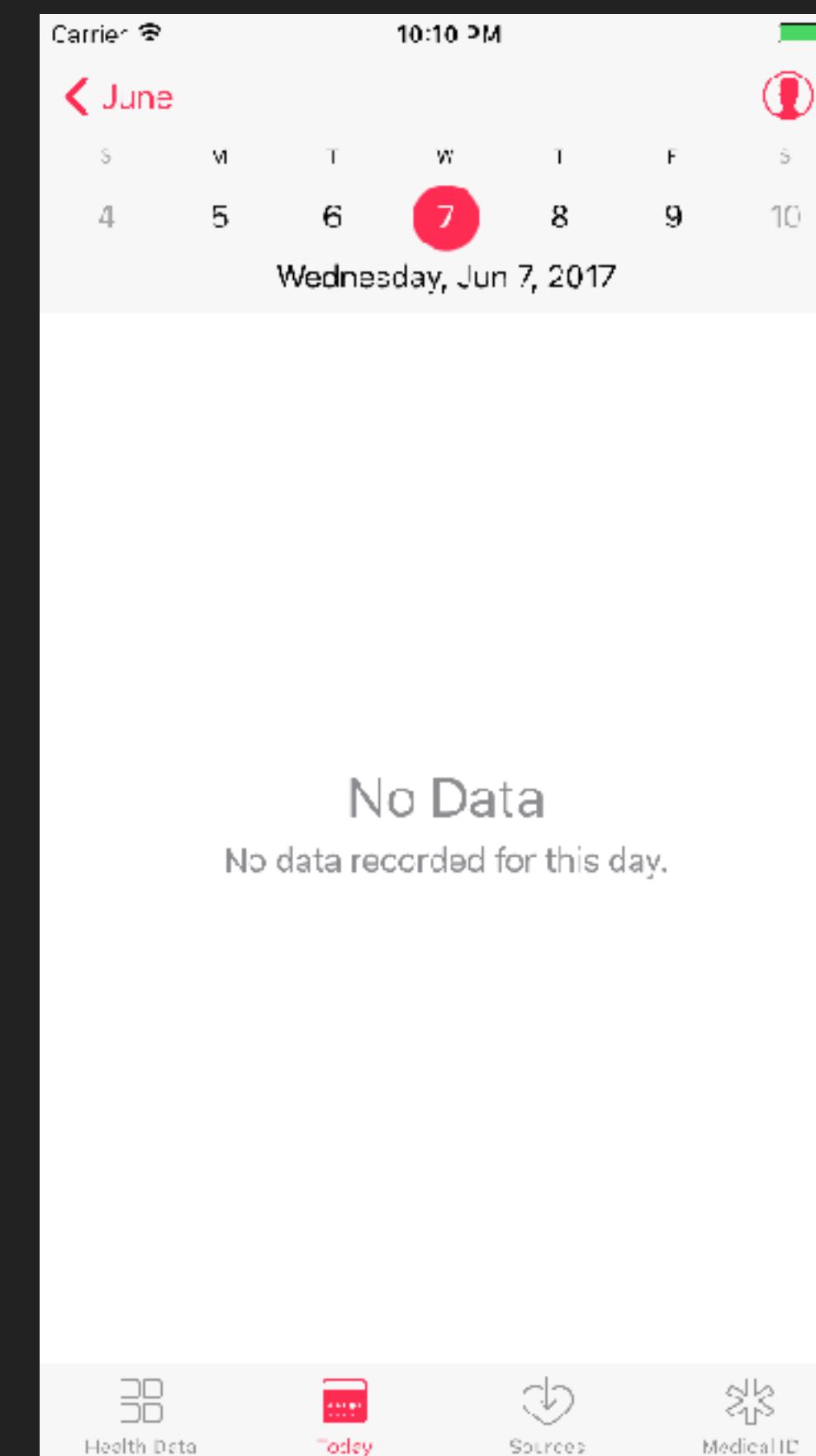
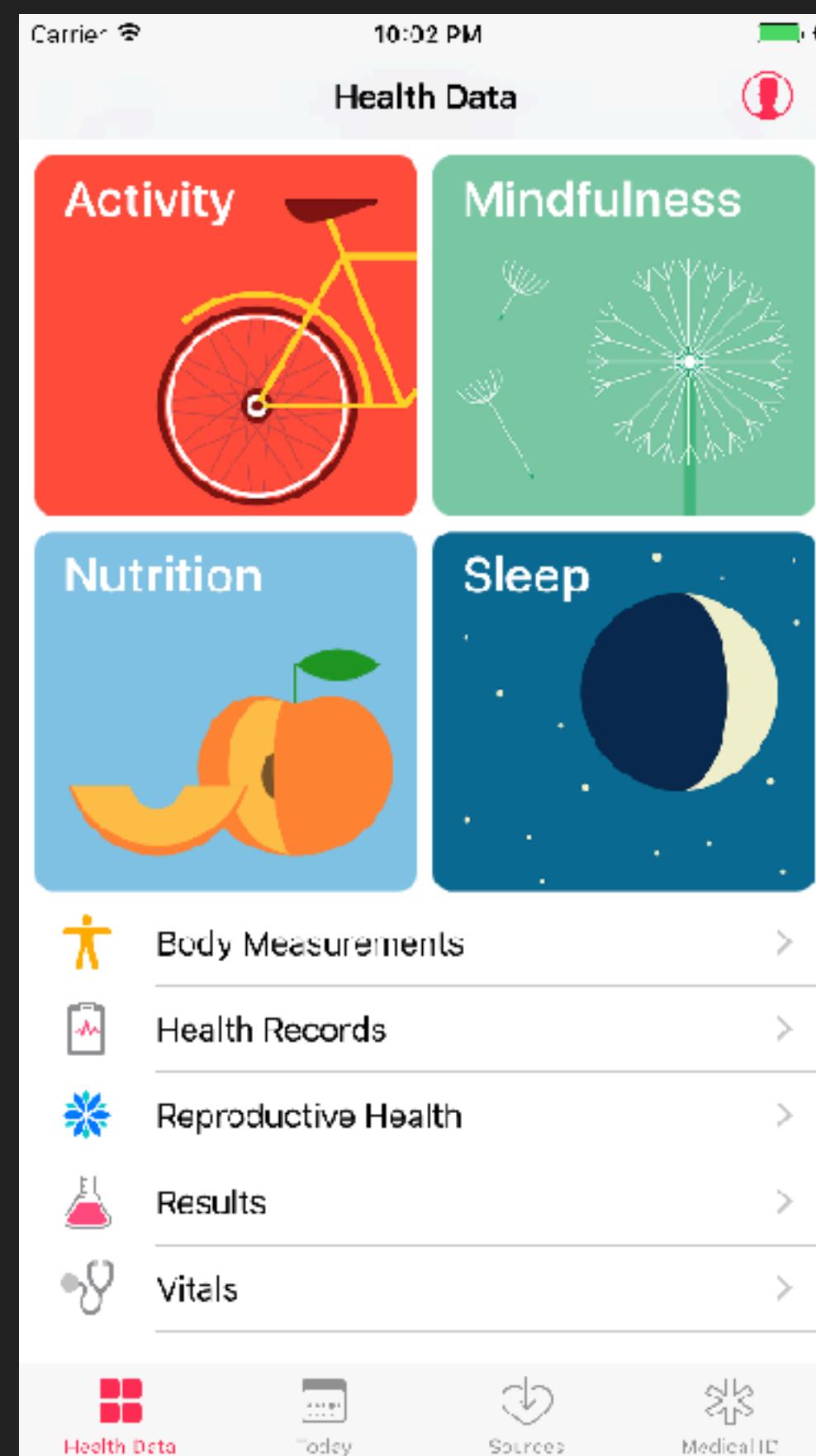
► Deixa você escolher entre MVCs diferentes...



ABAS!!!

UITABBARCONTROLLER

► Deixa você escolher entre MVCs diferentes...

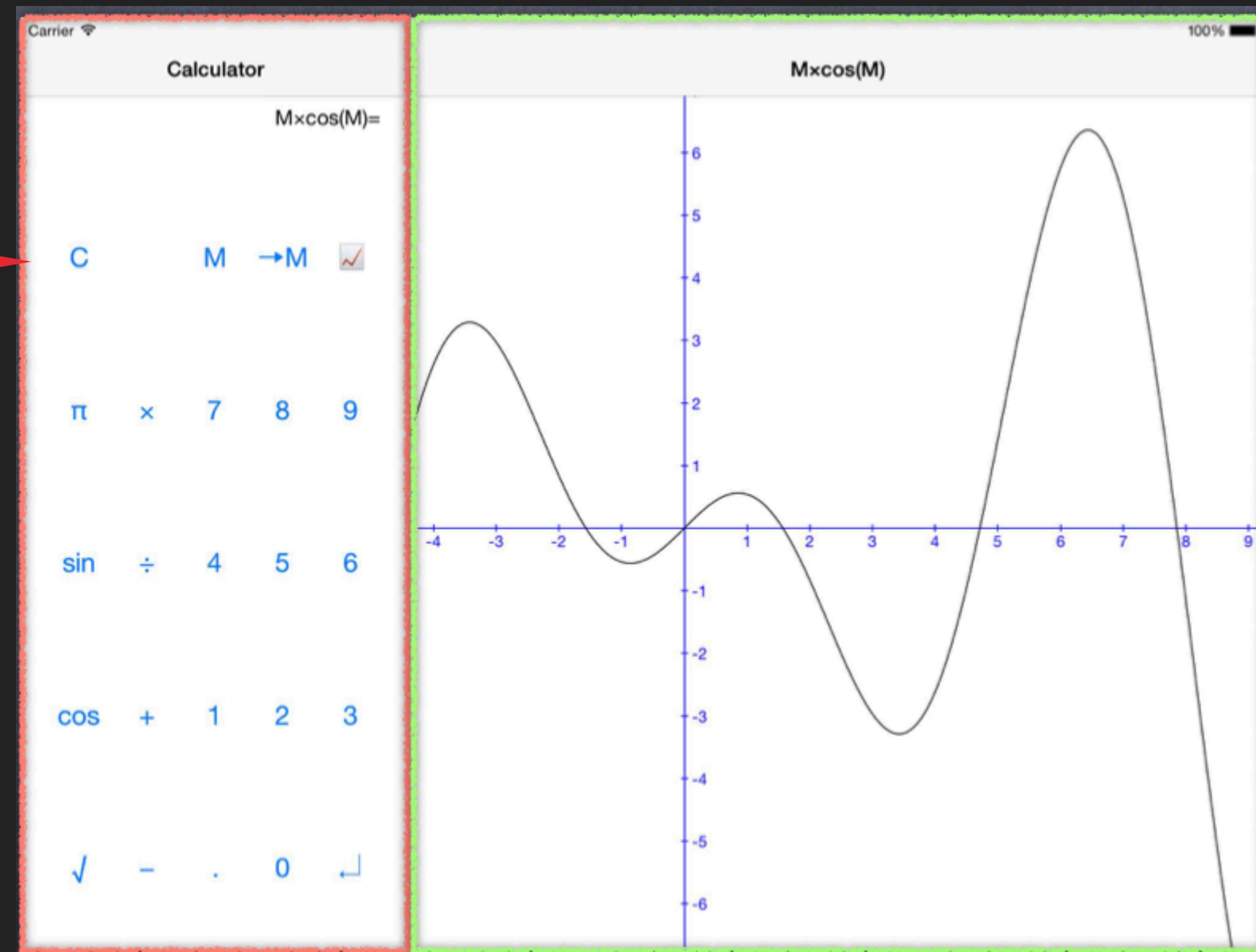


UISPLITVIEWCONTROLLER

- ▶ Coloca dois MVCs lado-a-lado...

UM MVC DE CALCULADORA

MASTER



UM MVC DE GRÁFICO

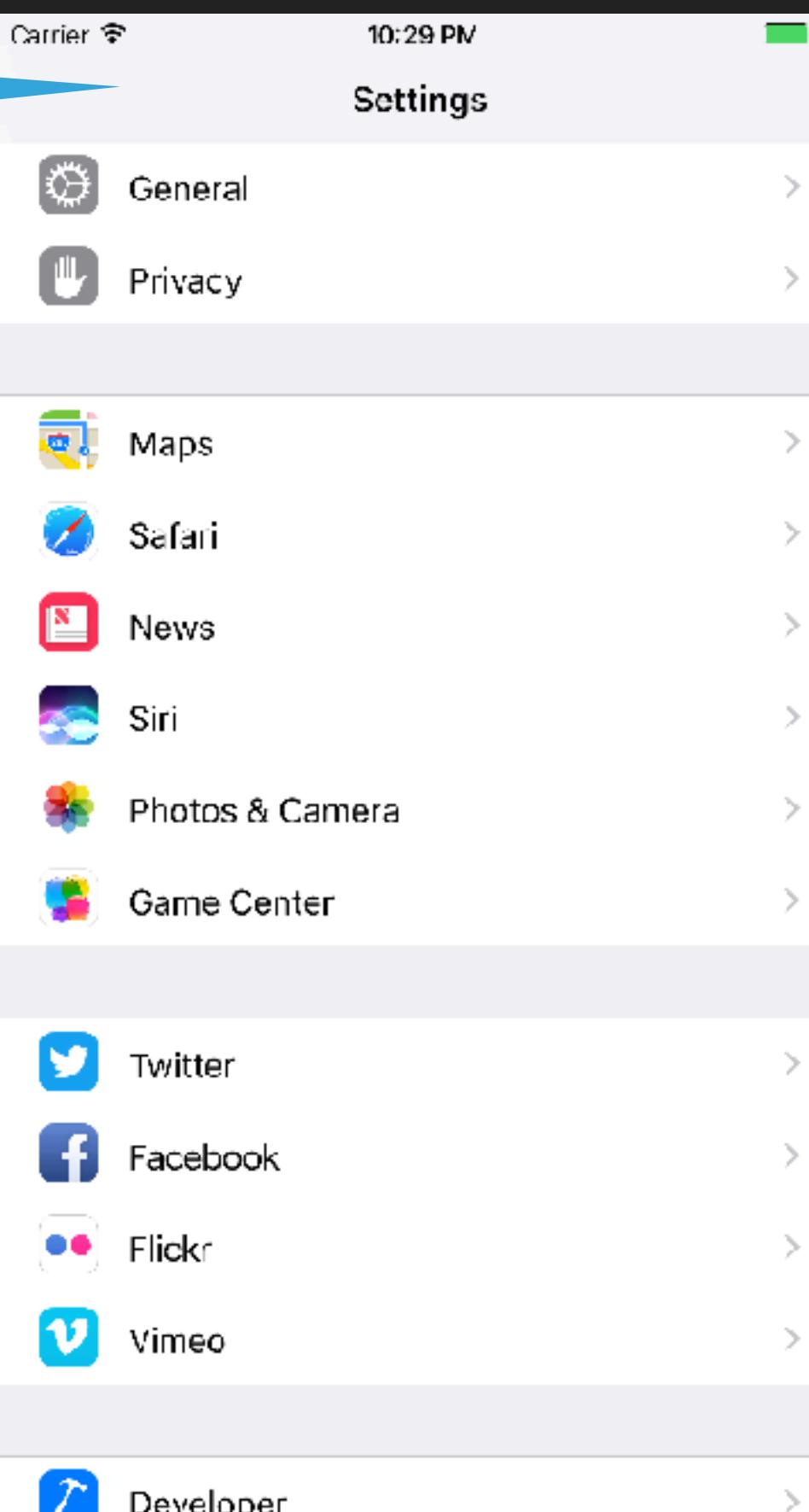
NAVEGAR É PRECISO, ENTÃO NÃO FIQUE AÍ À TOA!

UINavigationController

- ▶ Empilha outros MVCs, funcionando como uma pilha de cartas...

Esta área é desenhada pelo **UINavigationController**, mas o conteúdo dela (título, eventuais botões, etc) é determinado pelo MVC que está sendo exibido no momento, neste caso um MVC chamado **Settings**.

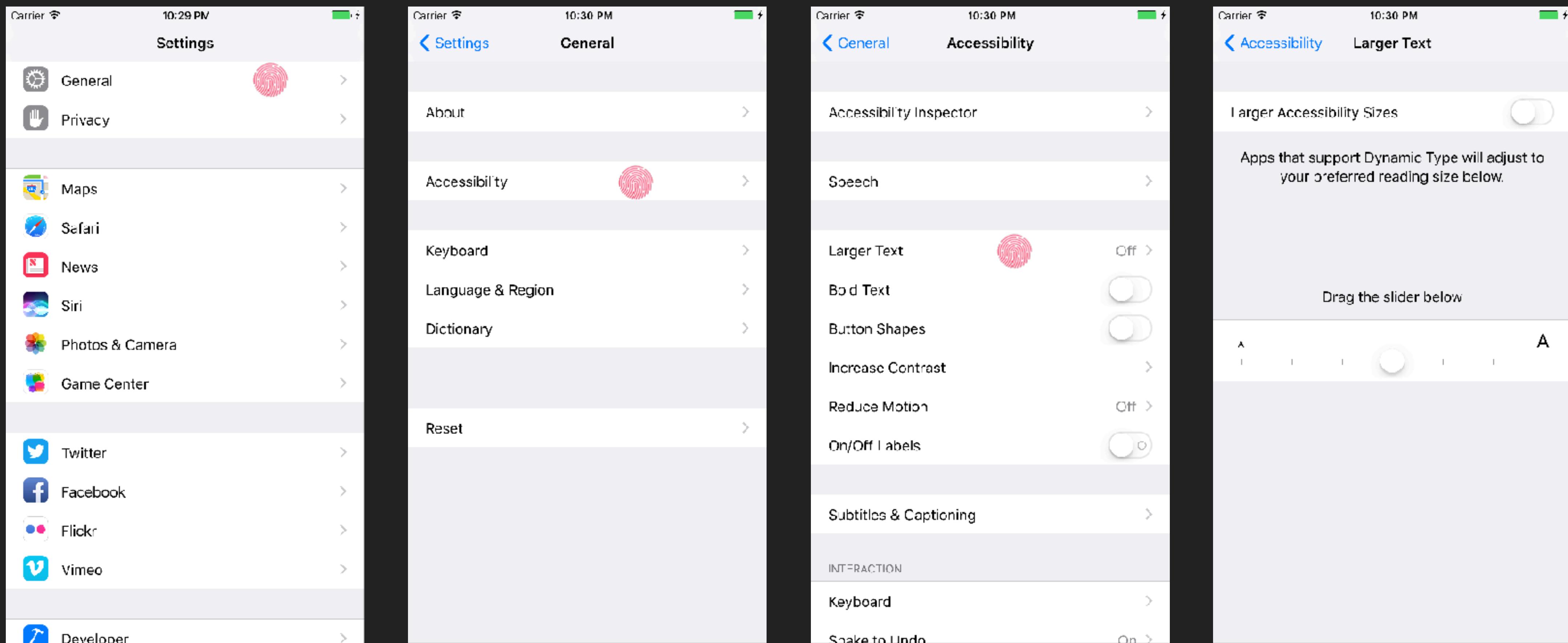
Cada MVC se comunica com esse conteúdo através da sua propriedade **navigationItem**.



NAVIGAR É PRECISO, ENTÃO NÃO FIQUE AÍ À TOA!

UINavigationController

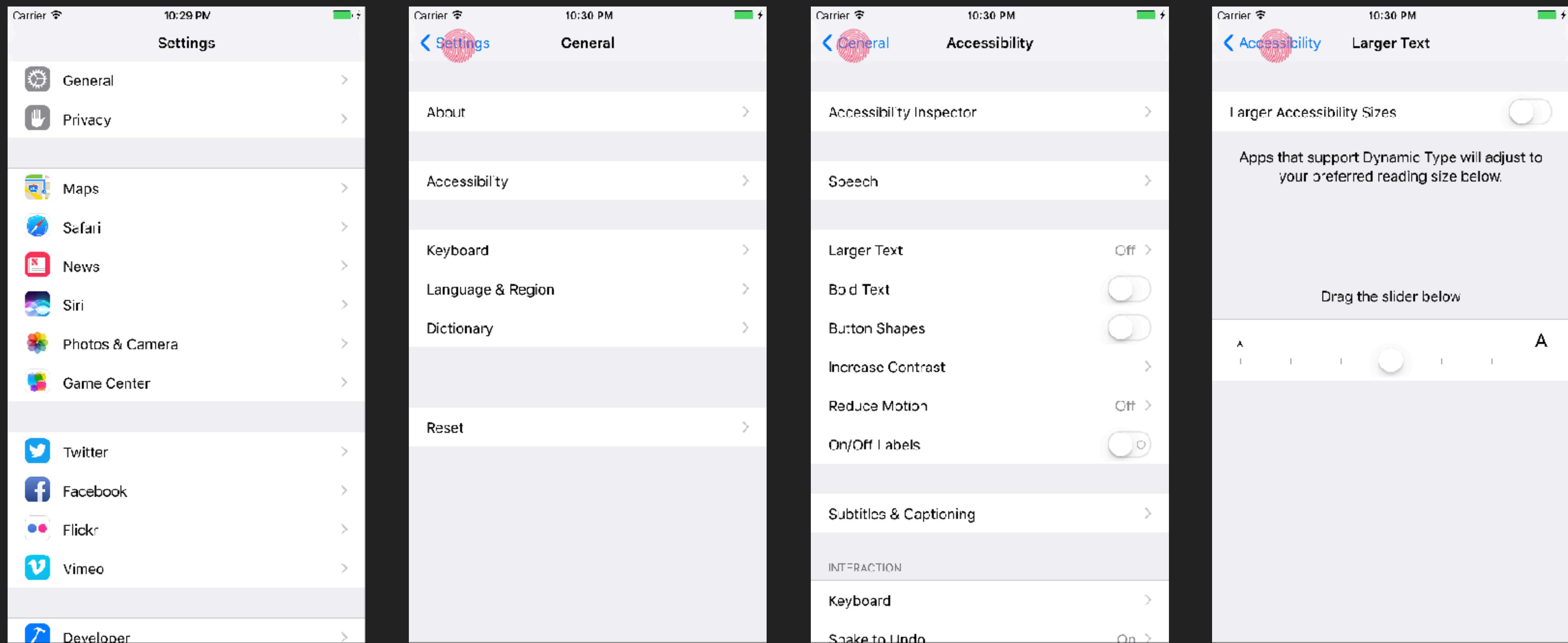
- ▶ Empilha outros MVCs, funcionando como uma pilha de cartas...



NAVIGAR É PRECISO, ENTÃO NÃO FIQUE AÍ À TOA!

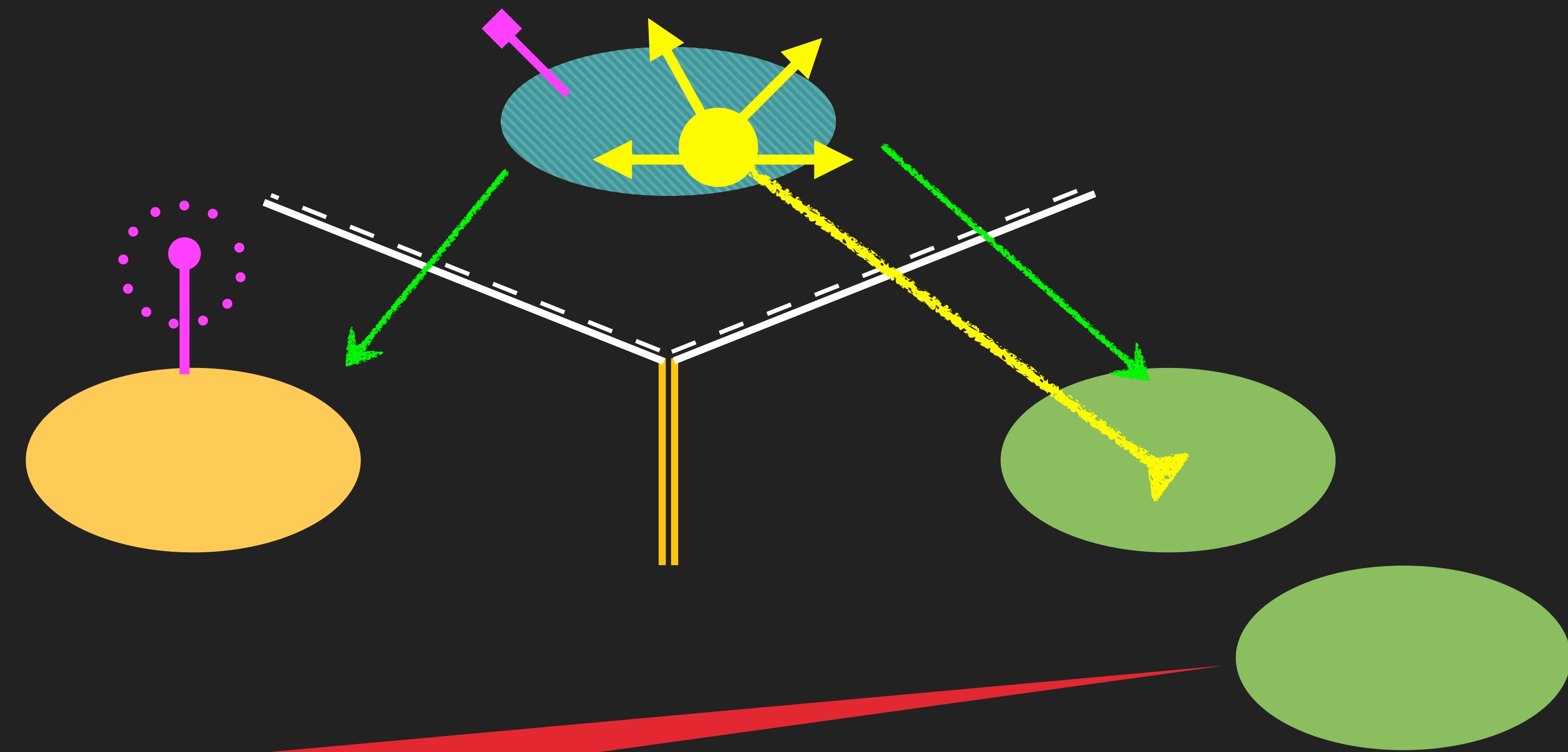
UINavigationController

- ▶ Empilha outros MVCs, funcionando como uma pilha de cartas...



IT'S THE RIME OF THE ANCIENT MARINER...

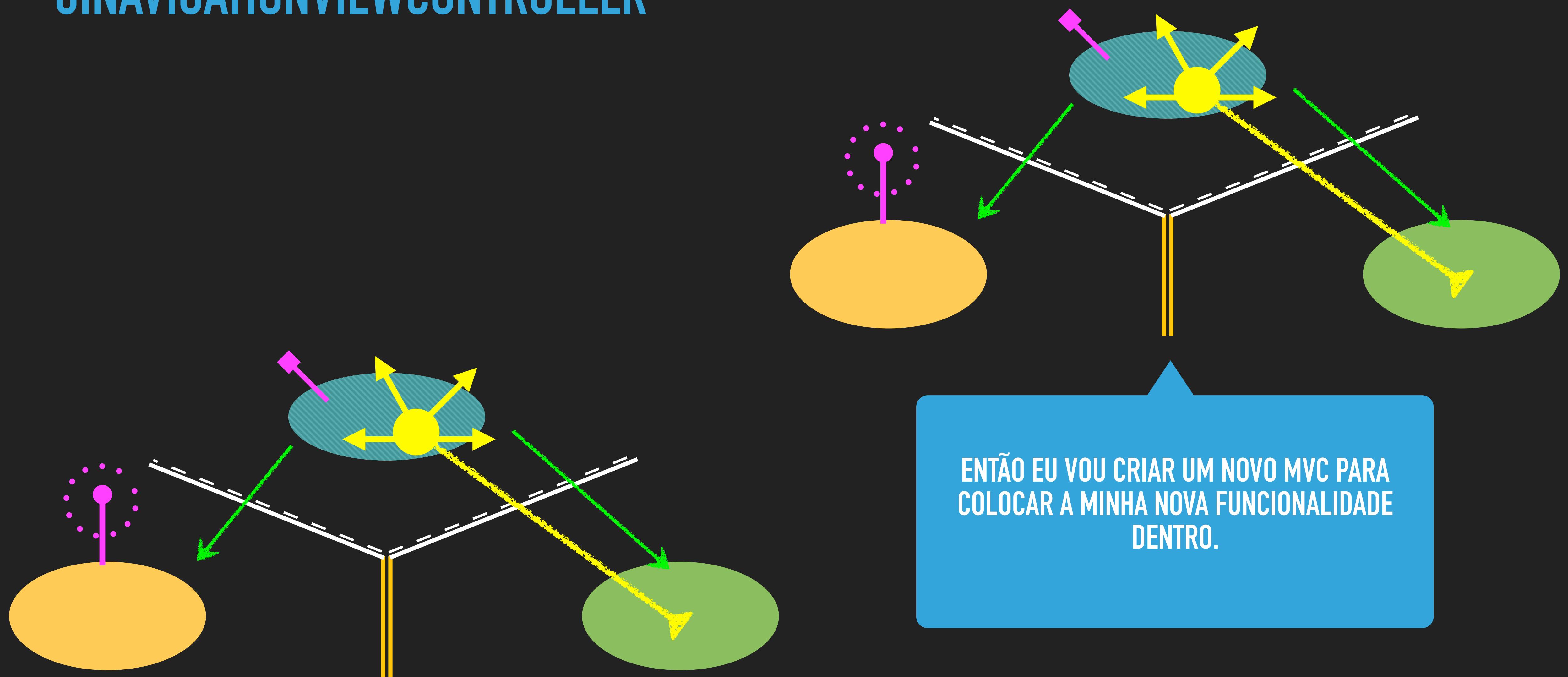
UINavigationViewController



EU QUERO MAIS FUNCIONALIDADES, MAS
NÃO FAZ SENTIDO COLOCÁ-LAS TODAS NUM
MESMO MVC!

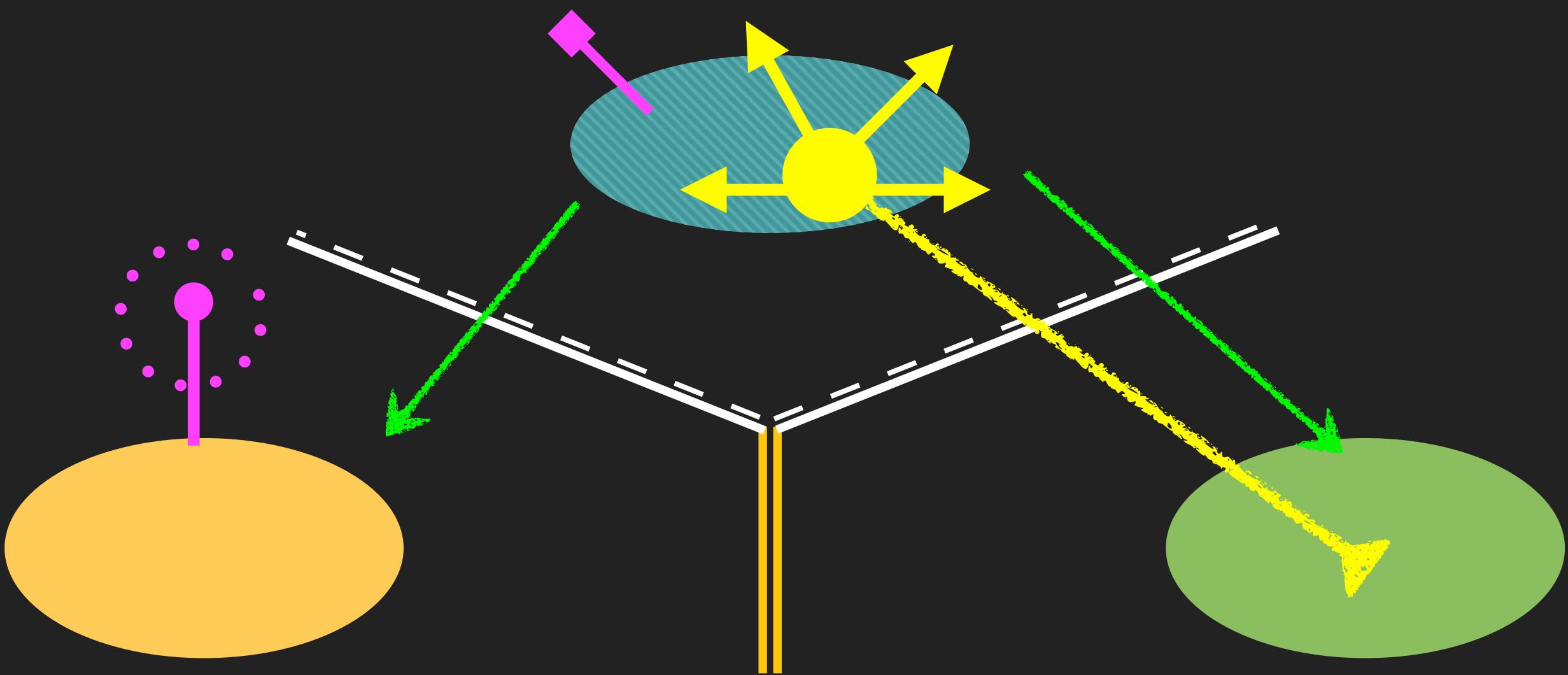
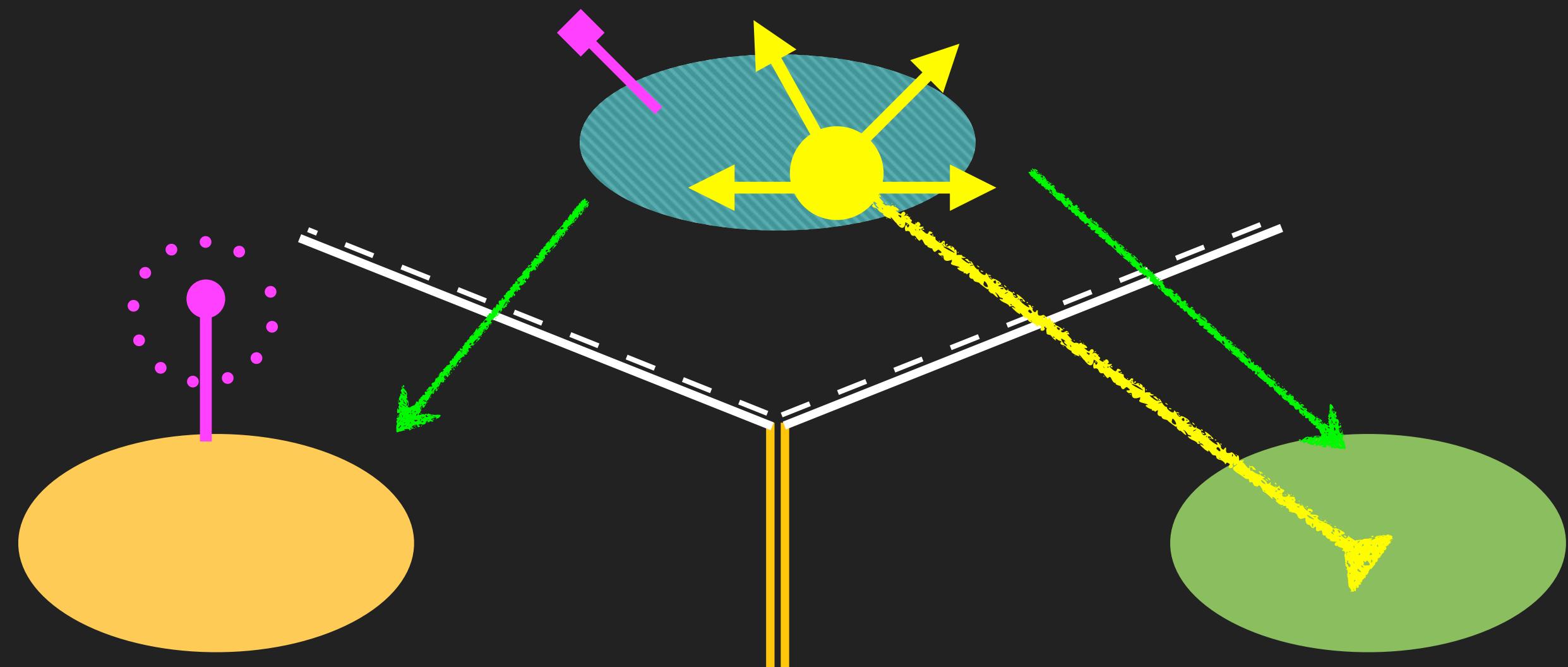
IT'S THE RIME OF THE ANCIENT MARINER...

UINavigationViewController



UINavigationViewController

Podemos usar o
UINavigationController para navegar entre
os nossos diversos MVCS.

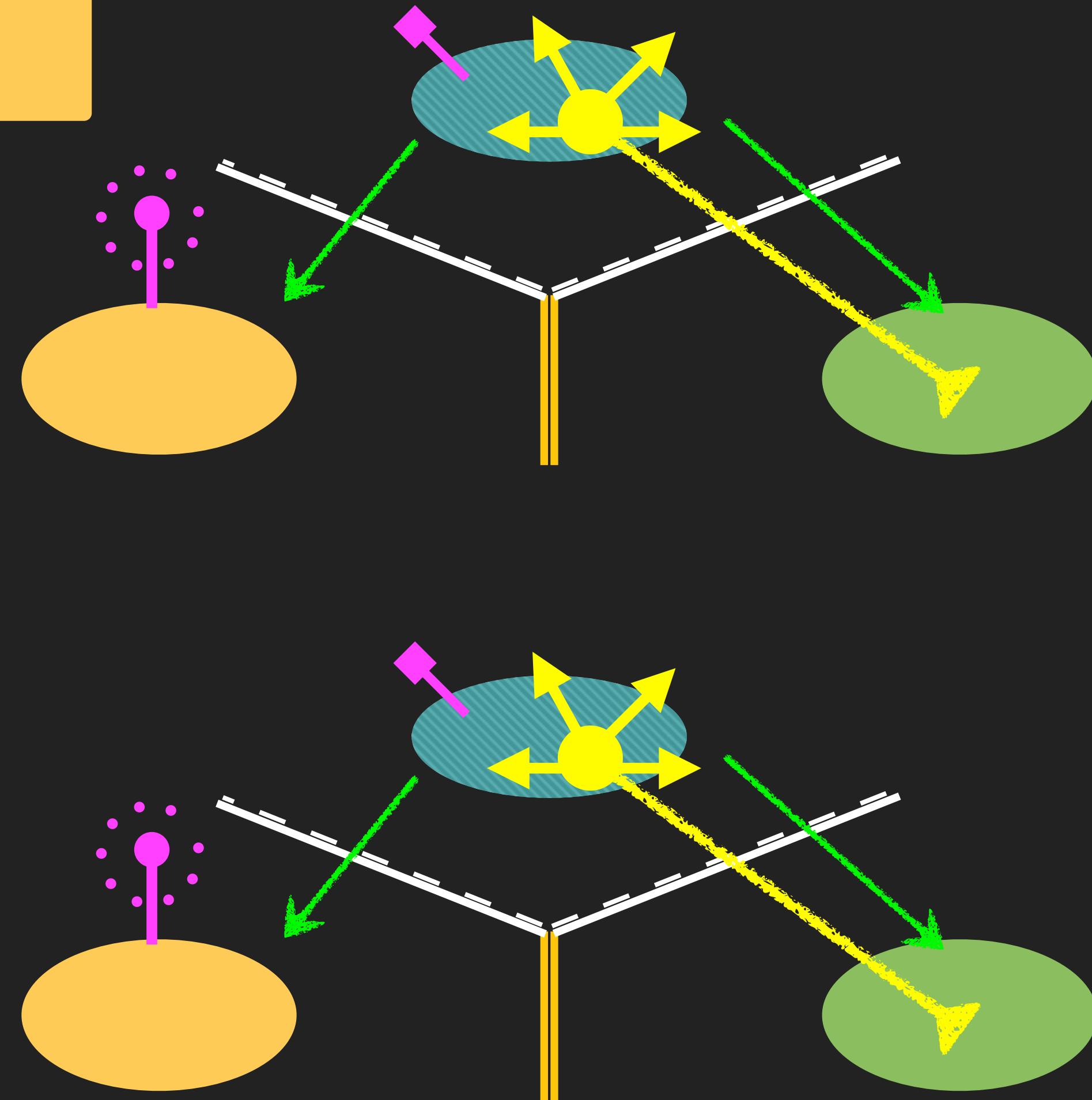
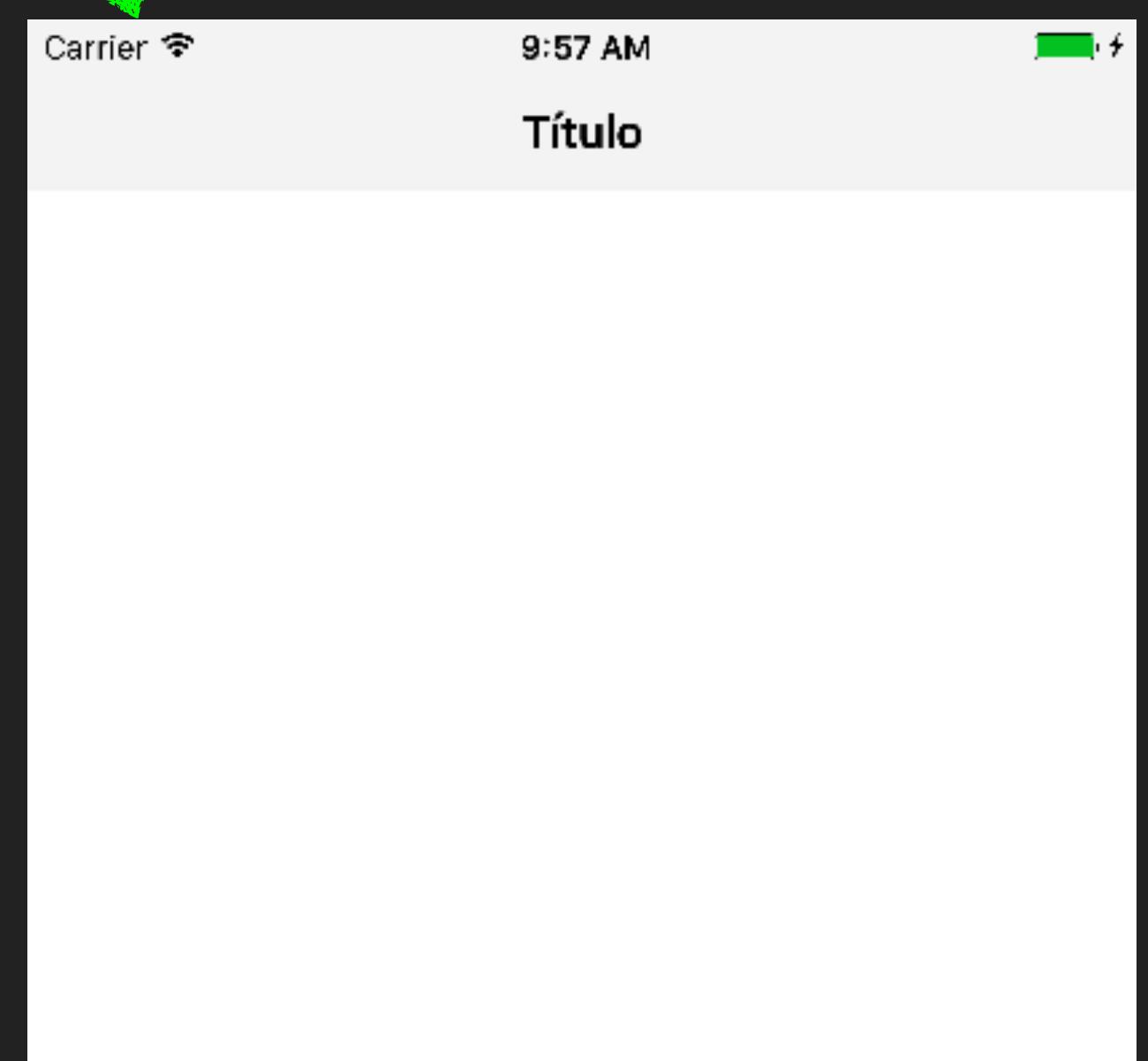


IT'S THE RIME OF THE ANCIENT MARINER...

UINavigationViewController

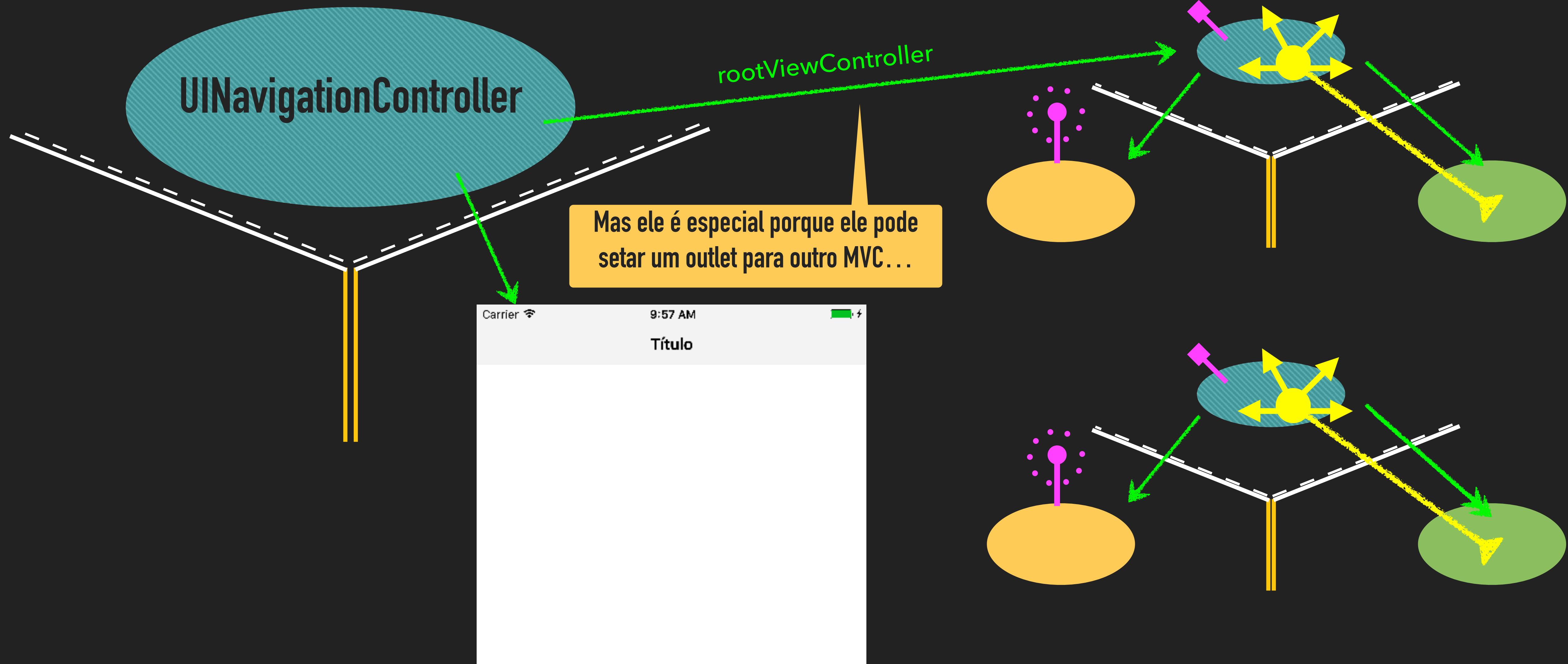
UINavigationController

Esta é a aparência do
UINavigationController.



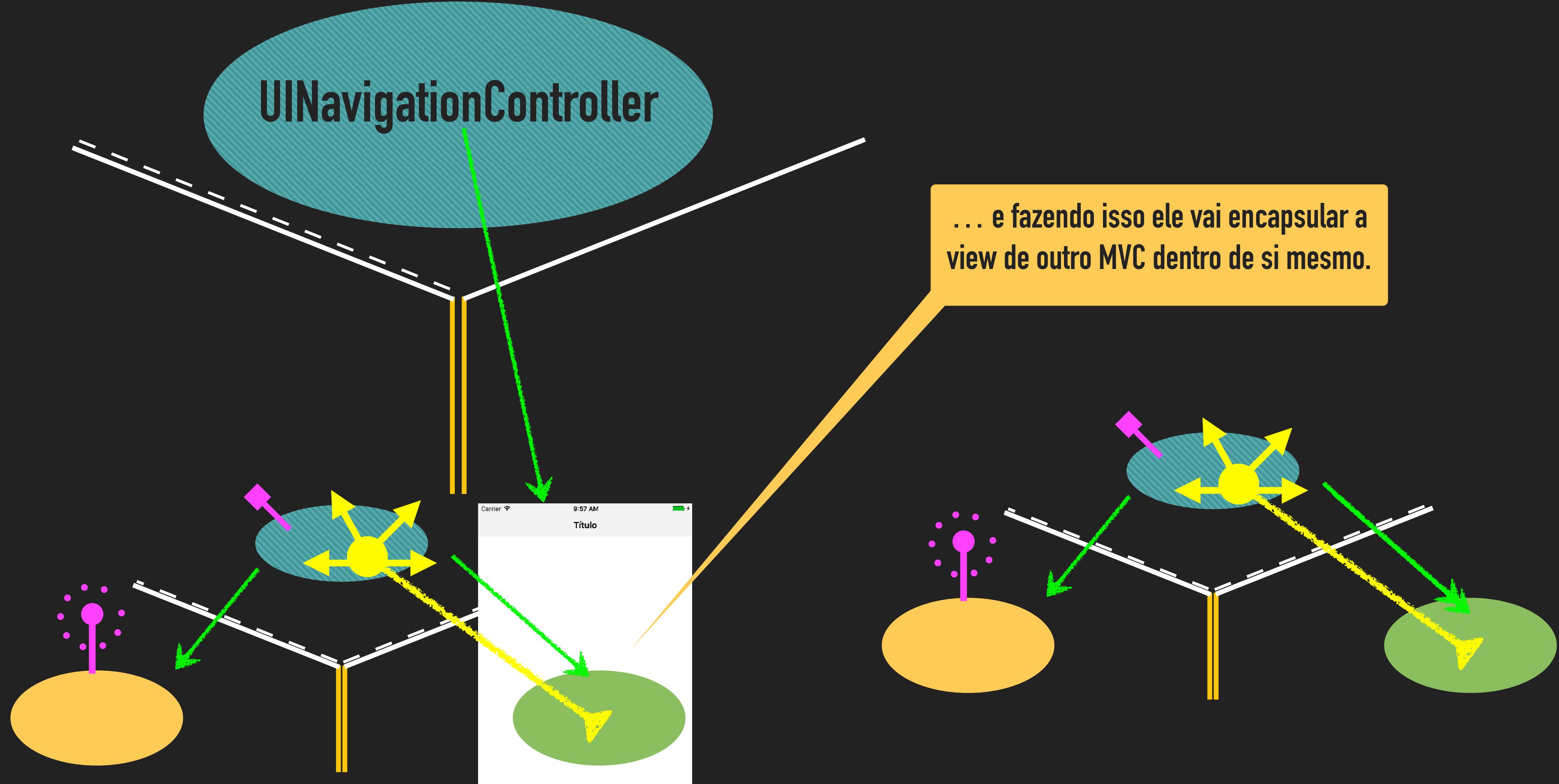
IT'S THE RIME OF THE ANCIENT MARINER...

UINavigationController



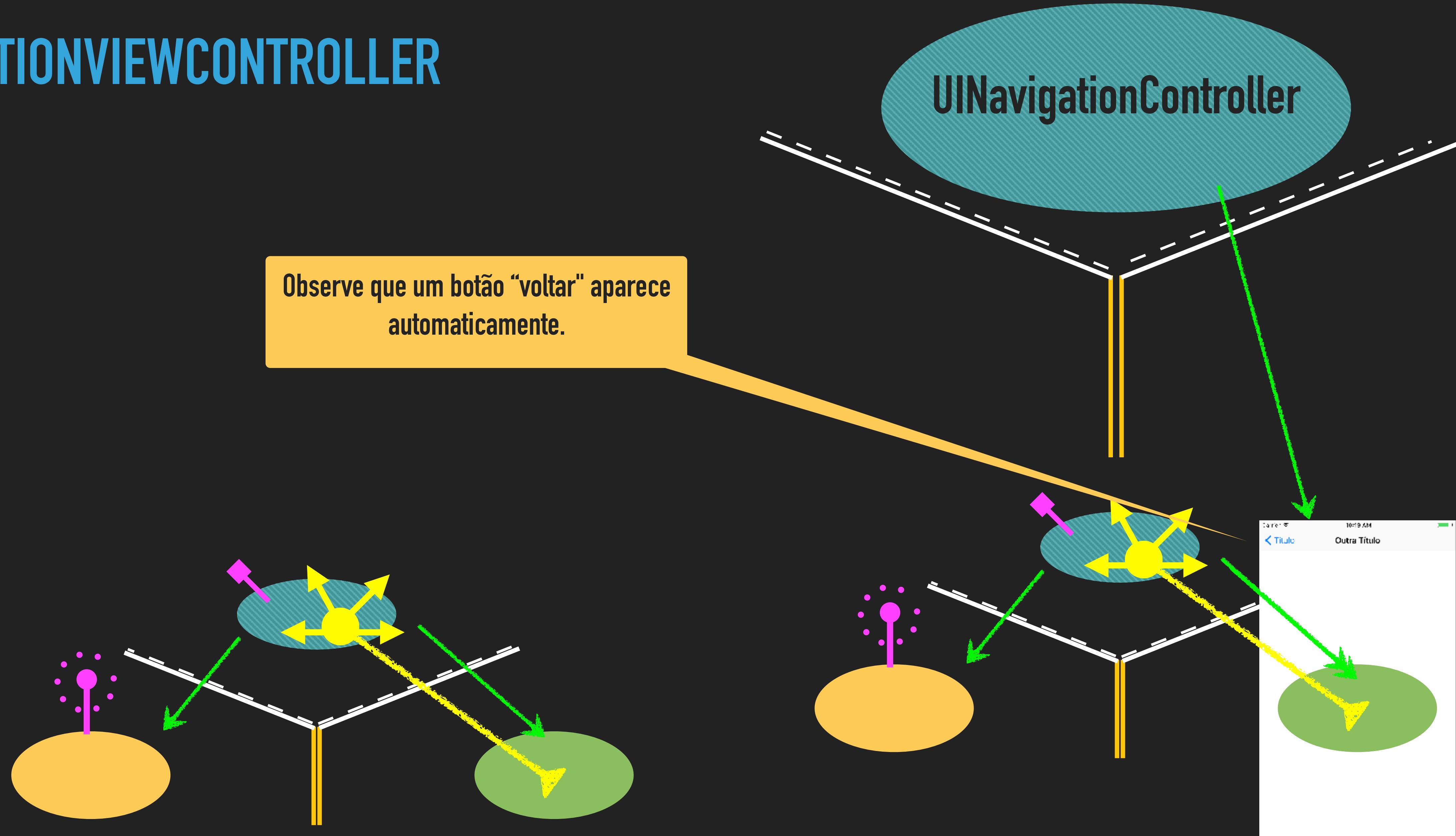
IT'S THE RIME OF THE ANCIENT MARINER...

UINavigationViewController



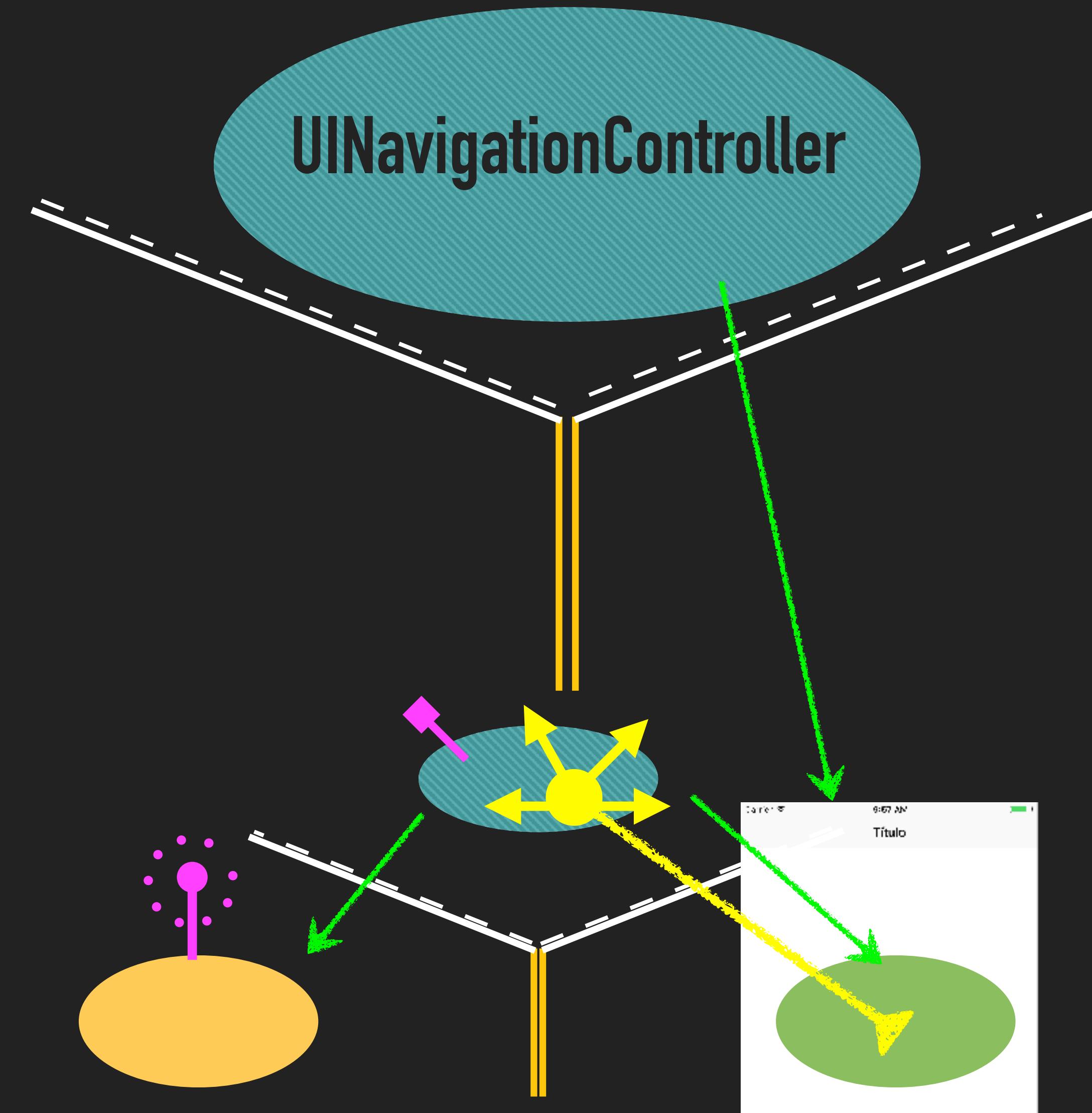
IT'S THE RIME OF THE ANCIENT MARINER...

UINavigationViewController



IT'S THE RIME OF THE ANCIENT MARINER...

UINavigationViewController



Perceba que quando usamos o botão voltar, o MVC anterior desaparece (ele é deslocado da memória).

ACESSANDO OS OUTROS MVCS

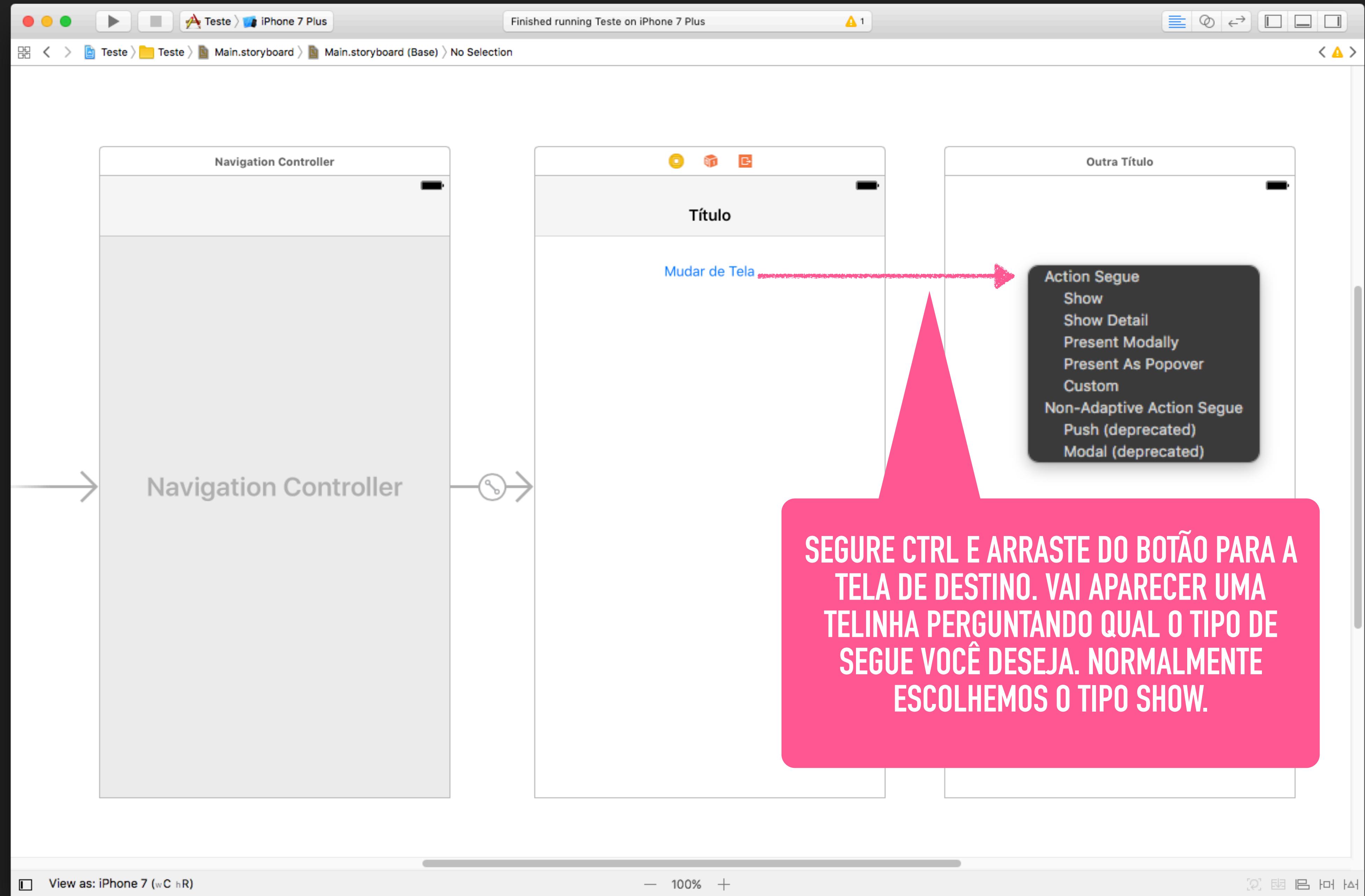
- ▶ Podemos obter os sub-MVCs do ViewController através da propriedade `viewControllers`
`var viewControllers: [UIViewController]? { get set }`
- ▶ Para o Tab Bar, os MVCs estão na ordem das abas (da esquerda para a direita)
- ▶ Para o Split View, a posição 0 é o master e a posição 1 é o detail
- ▶ Para o Navigation Controller, a posição 0 é o `rootViewController` e o resto segue a ordem de empilhamento
- ▶ Muito embora esta variável seja aberta, normalmente o conteúdo dela é atribuído através do storyboard, **segues** ou ainda através do empilhamento da navegação.

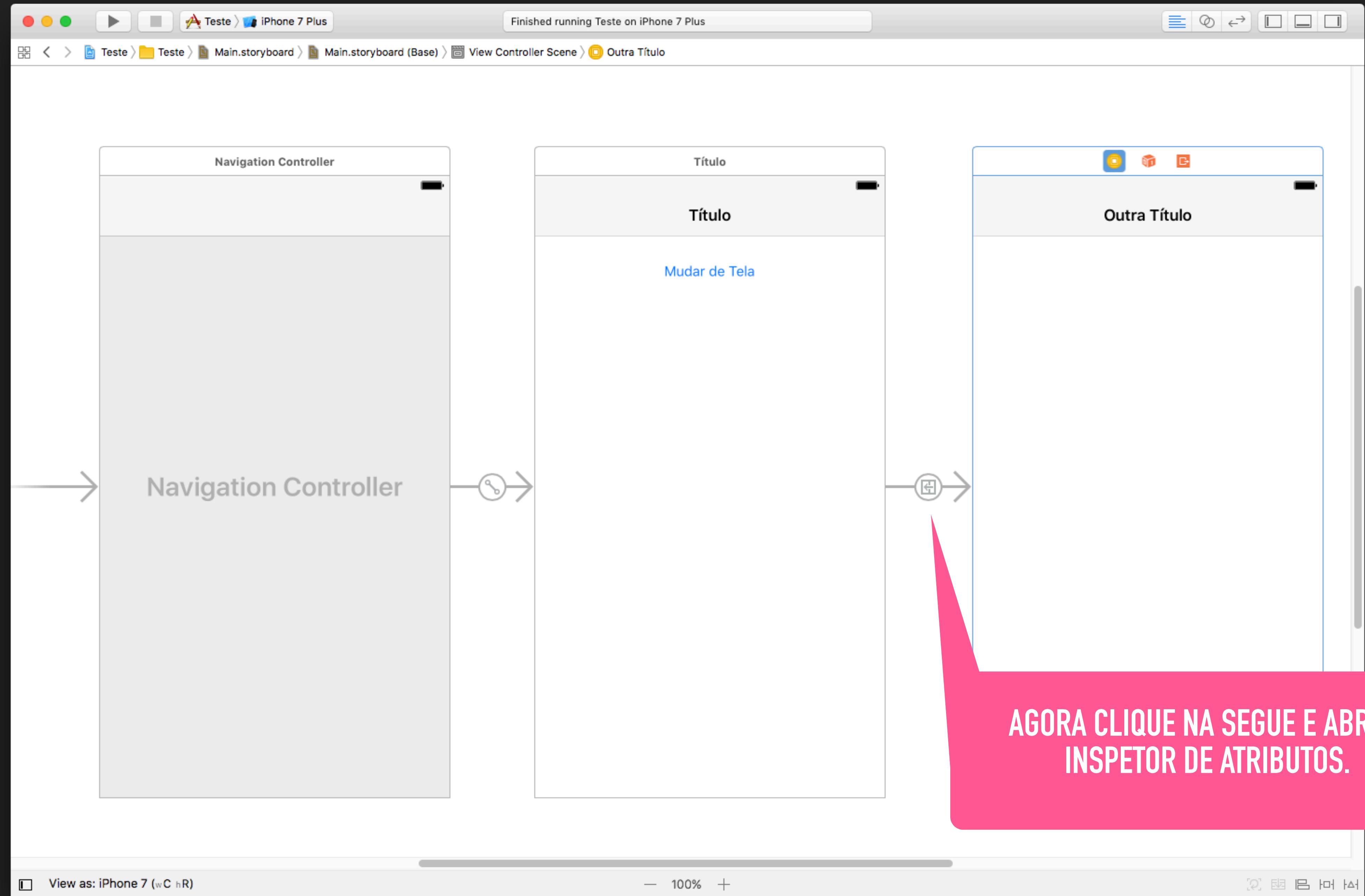
SEGUES

- ▶ Agora que temos nossos exemplos, cada qual com uma única tela, e agora?
- ▶ Se quisermos criar algo mais complexo, com mais telas (mais de um MVC), o que vamos usar para trocar de uma tela para a outra?
- ▶ Usamos uma coisa chamada **segue!**
- ▶ As **segues** se adaptam ao ambiente, através dos tipos:
 - show - vão provocar um empilhamento num `UINavigationController`, se um não existir, a nova tela entrará como Modal
 - show detail - mostra o detail num `UISplitViewController` (pode fazer empilhamento também)
 - modal - auto-explicativa
 - popover - faz o novo MVC aparecer num popover (parece com um popup)

SEGUE

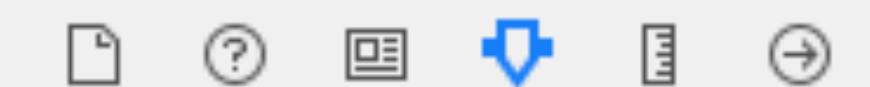
- ▶ Uma **segue** sempre cria uma nova instância do MVC de destino;
- ▶ Mesmo quando estamos lidando com master/detail, ambos podem ser trocados por uma nova instância
- ▶ Quando você faz uma **segue** que provoca empilhamento num UINavigationController, a tela de destino será uma nova instância. **Importante:** o ato de voltar não representa uma **segue**, portanto a tela anterior não será trocada por uma nova instância
- ▶ Como fazer essa magia de segue, afinal?
Muito simples! Segure CTRL e arraste de um botão para a nova tela e pronto!





Finished running Teste on iPhone 7 Plus

Controller Scene > Show segue to "Outra Título"



Storyboard Segue

Identifier	Identifier
Class	UIStoryboardSegue
Module	None
<input type="checkbox"/> Inherit From Target	
Kind	Show (e.g. Push)
<input checked="" type="checkbox"/> Animates	
Peek & Pop	<input type="checkbox"/> Preview & Commit Segues

DÊ PARA A SUA SEGUE UM IDENTIFICADOR.
ELE DEVE SER DESCRIPTIVO A RESPEITO DO
QUE FAZ AQUELA SEGUE.

Título

Mudar de Tela



IDENTIFICADOR DA SEGUE

- ▶ Serve para ser usado no código em dois principais momentos:
 - ▶ Quando você quiser evocar aquela segue programáticamente usando o método `performSegue(withIdentifier: String, sender: Any?)`
 - ▶ Ou quando você vai implementar um método especial para se preparar para a mudança de tela - este é o uso mais comum e o mais importante!
- ▶ Quando uma segue acontece, o `ViewController` que contém o objeto que iniciou a segue (o botão do nosso exemplo) tem a chance de se preparar e preparar o destino para ser iniciado.
- ▶ **Lembre-se**, mais uma vez, que a MVC de destino é sempre uma nova instância (nunca há reuso de instâncias neste caso)

PREPARANDO PARA UMA SEGUE

- ▶ Este método é chamado dentro do ViewController que possui o componente que provocou a segue:

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {  
    if let identifier = segue.identifier {  
        if identifier == "vaiParaTela2" {  
            if let destination = segue.destination as? SecondViewController {  
                //setar propriedades, chamar métodos, etc...  
                //IMPORTANTE: isto ocorre antes do viewDidLoad()  
                // falaremos do ciclo de vida em breve.  
            }  
        }  
    }  
}
```

PREPARAR, APONTAR, FOGO!!!

PREPARANDO PARA UMA SEGUE

O IDENTIFICADOR QUE COLOCAMOS NO STORYBOARD

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {  
    if let identifier = segue.identifier {  
        if identifier == "vaiParaTela2" {  
            if let destination = segue.destination as? SecondViewController {  
                //setar propriedades, chamar métodos, etc...  
                //IMPORTANTE: isto ocorre antes do viewDidLoad()  
                // falaremos do ciclo de vida em breve.  
            }  
        }  
    }  
}
```

COMPARAMOS O IDENTIFICADOR RECEBIDO
COM O QUE QUEREMOS, PARA ENTÃO
FAZERMOS A NOSSA LÓGICA

SENDER - É O OBJETO QUE DEU ORIGEM À SEGUE OU UM
OBJETO FORNECIDO PROGRAMATICAMENTE, QUANDO A
SEGUE É FEITA DESTA MANEIRA.

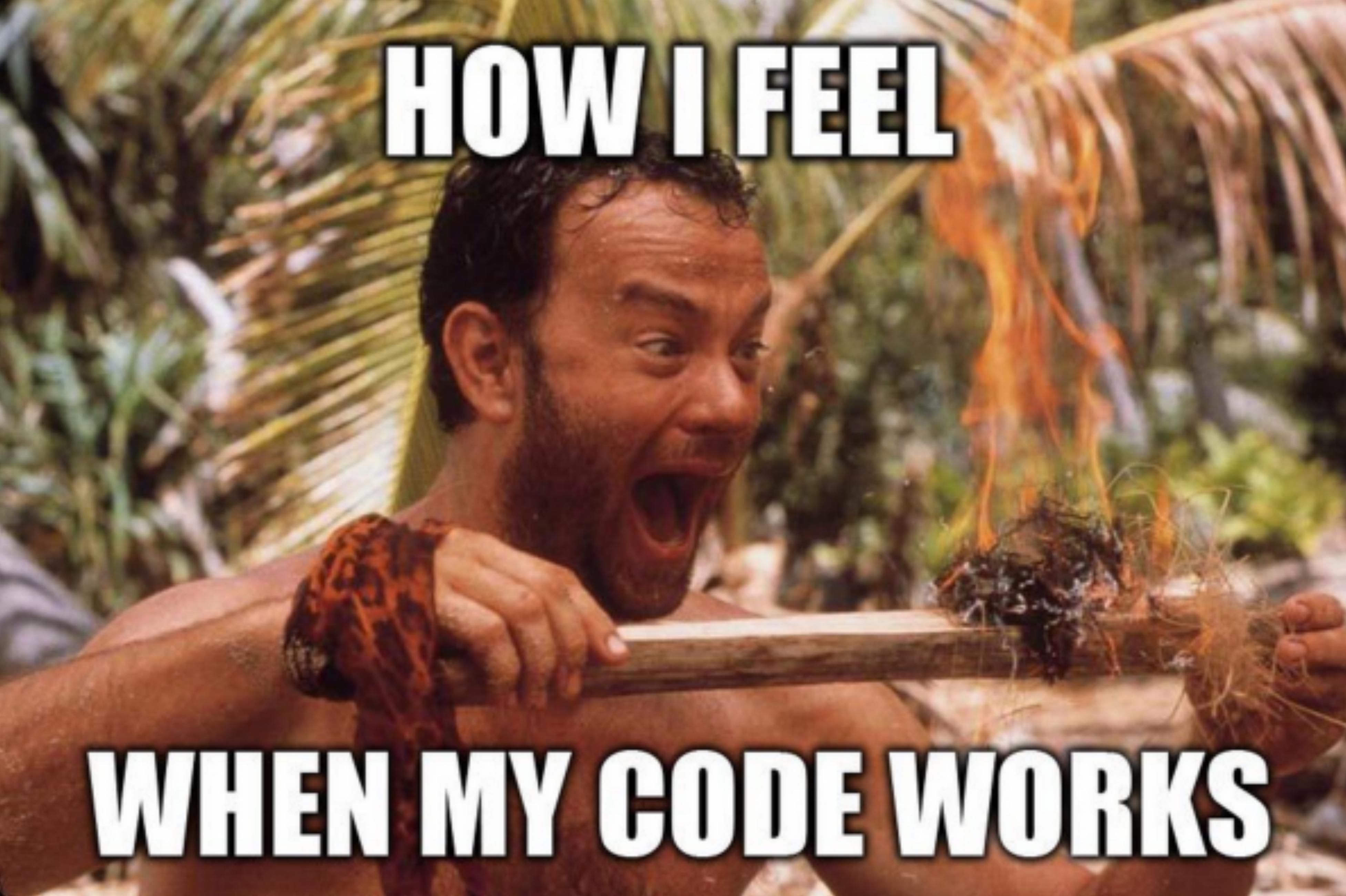
OBTENDO A INSTÂNCIA DO VIEWCONTROLLER DE
DESTINO E FAZENDO O CAST PARA A CLASSE
ADEQUADA

ATENÇÃO!!! ESTA PREPARAÇÃO PARA SEGUES ACONTECE ANTES DOS OUTLETS SEREM SETADOS!

NÃO! EU SOU RYYYYCAAAAAAAA!

IMPEDINDO SEGUES DE ACONTECER

- ▶ Existe ainda a possibilidade de impedirmos uma segue de acontecer, baseado na nossa lógica de negócio
- ▶ Para fazer isso, basta retornar **false** no método
shouldPerformSegue(withIdentifier: String? sender: Any?) -> Bool
- ▶ O **identifier** é aquele que configuramos no Storyboard;
- ▶ O **sender** é o objeto que provocou a segue (ex.: um botão)

A photograph of a man with dark skin and short hair, wearing a patterned orange and black shirt. He is shouting with his mouth wide open, showing his teeth. He is leaning over a low wooden platform or table. In front of him is a small fire with bright orange flames. The background is filled with dense green tropical foliage and palm fronds. The overall atmosphere is one of intense emotion and energy.

HOW I FEEL

WHEN MY CODE WORKS

DE POSSE DO EXERCÍCIO DA AULA
ANTERIOR, COLOQUE CADA VIEW NUMA
ABA DE UM UITABBARCONTROLLER

Professor Pedro Henrique

K-BÔ