

# WebView

Utilizando StoryBoard

X-Code com Swift

Prof. Agesandro Scarpioni

[agesandro@fiap.com.br](mailto:agesandro@fiap.com.br)



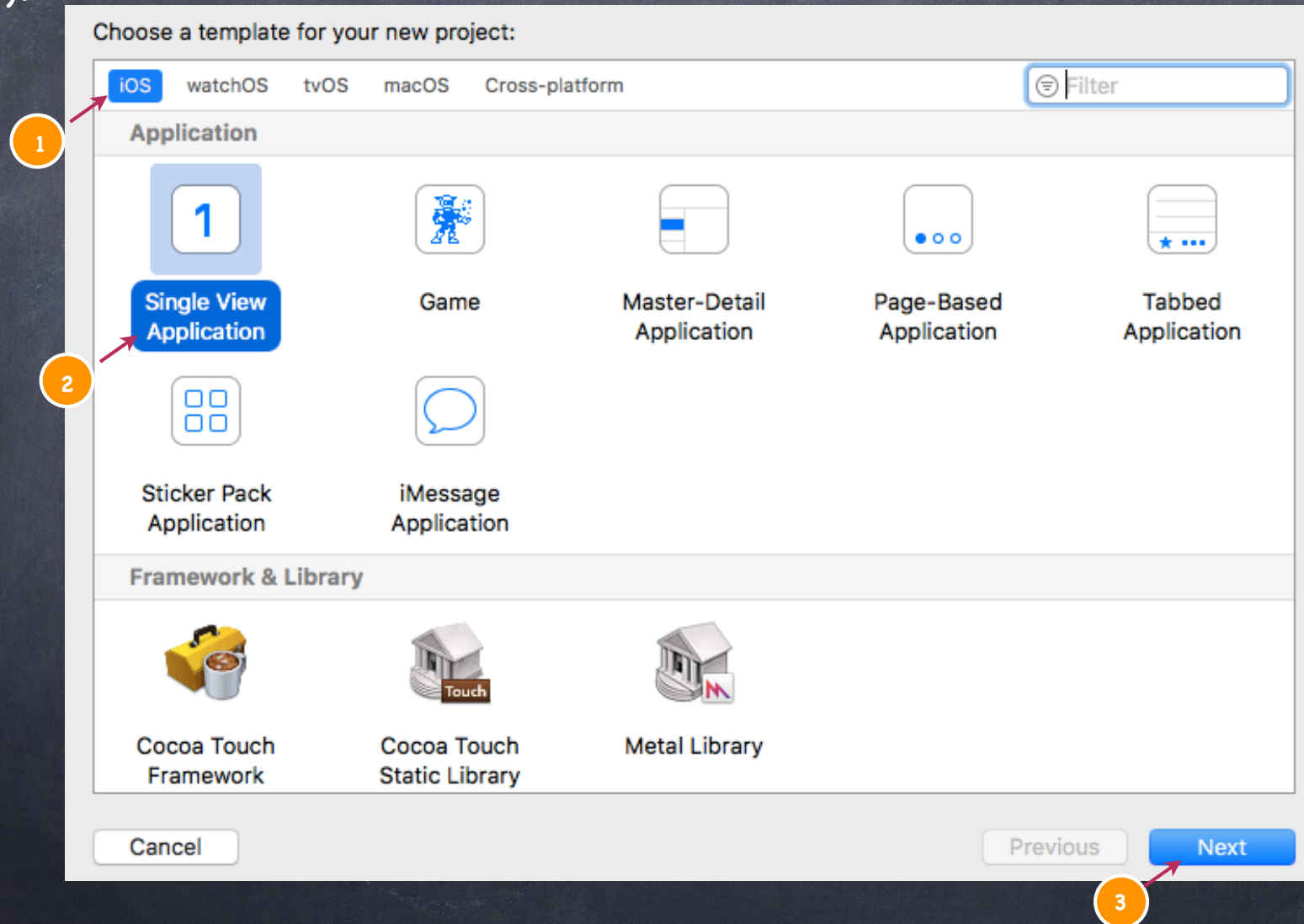
# WebView

- Vamos ver como é fácil utilizar o WebView, um objeto muito utilizado para exibir páginas Web juntamente com outro objeto chamado activity.



# WebView

- Crie um projeto novo do tipo iOS application(1), Single View Application(2) clique em Next(3).



# WebView

- Nomeie o projeto como: "Exemplo1\_WebView\_Swift" (4), selecione a linguagem Swift(5) e o device iPhone(6).

Choose options for your new project:

Product Name:

Team:

Organization Name:

Organization Identifier:

Bundle Identifier:

Language:

Devices:

☐ Use Core Data

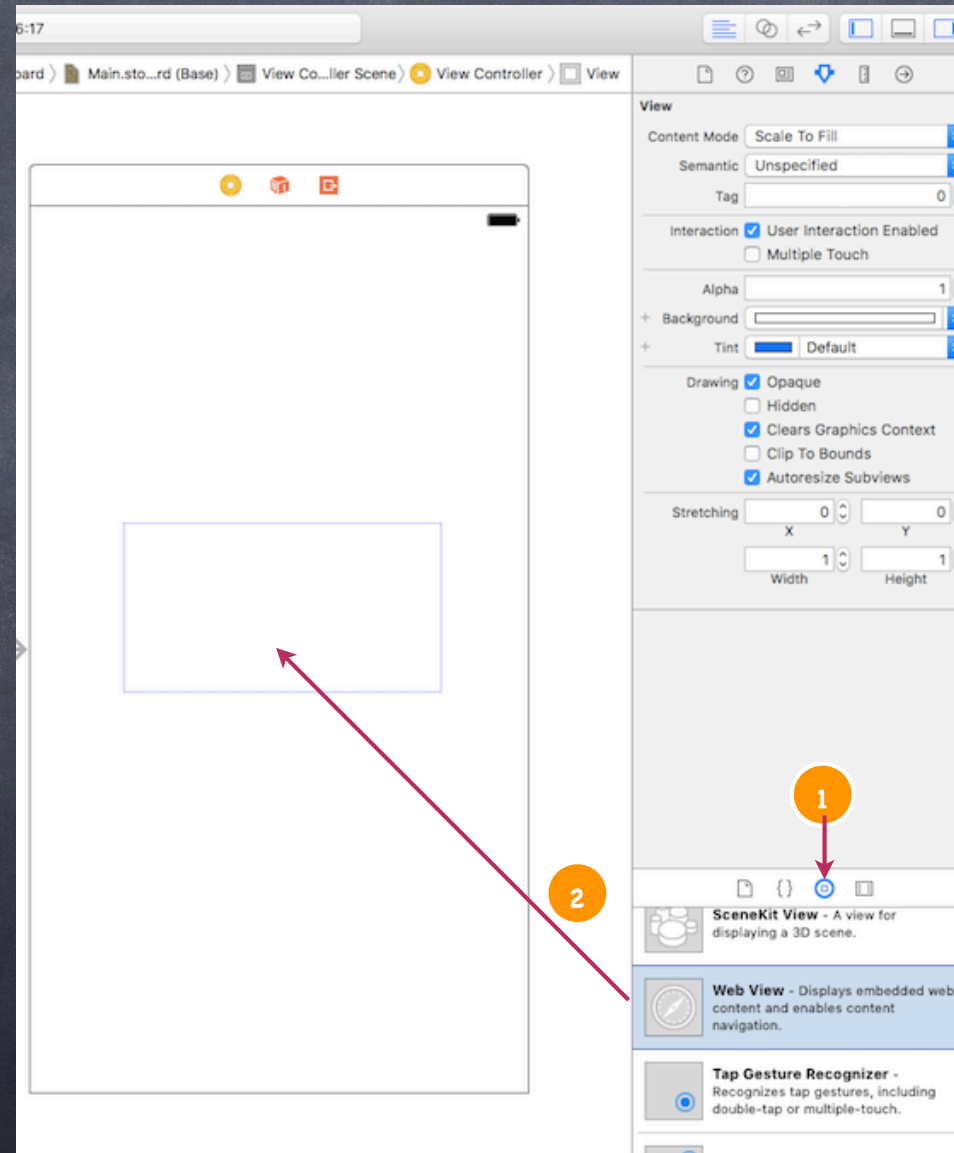
☐ Include Unit Tests

☐ Include UI Tests



# WebView

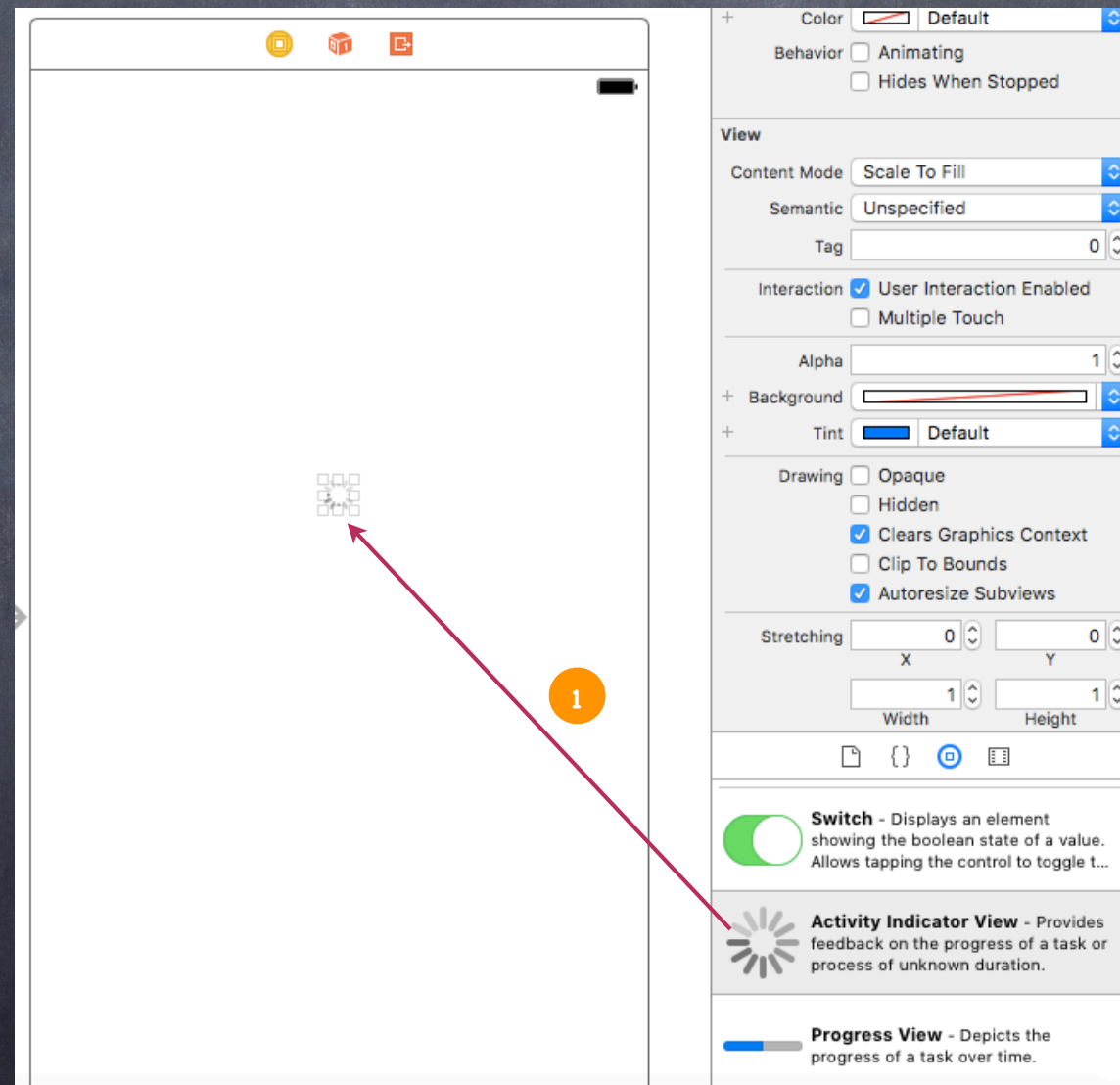
- Insira um objeto WebView em nossa View, basta arrastá-lo, depois estique o objeto para ocupar toda a área da View.





# WebView

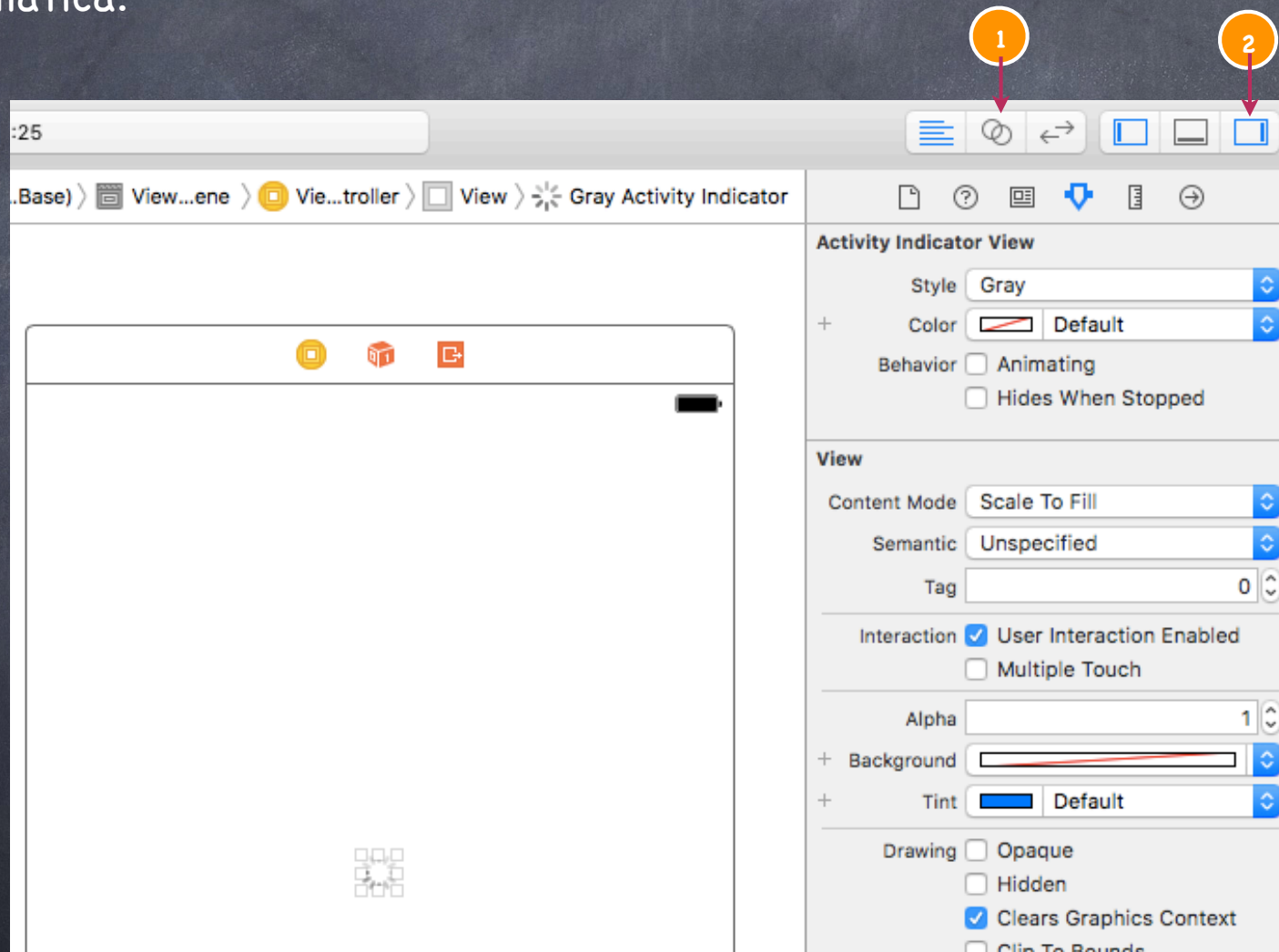
- Insira um objeto Activity Indicator View (1) sobre a WebView





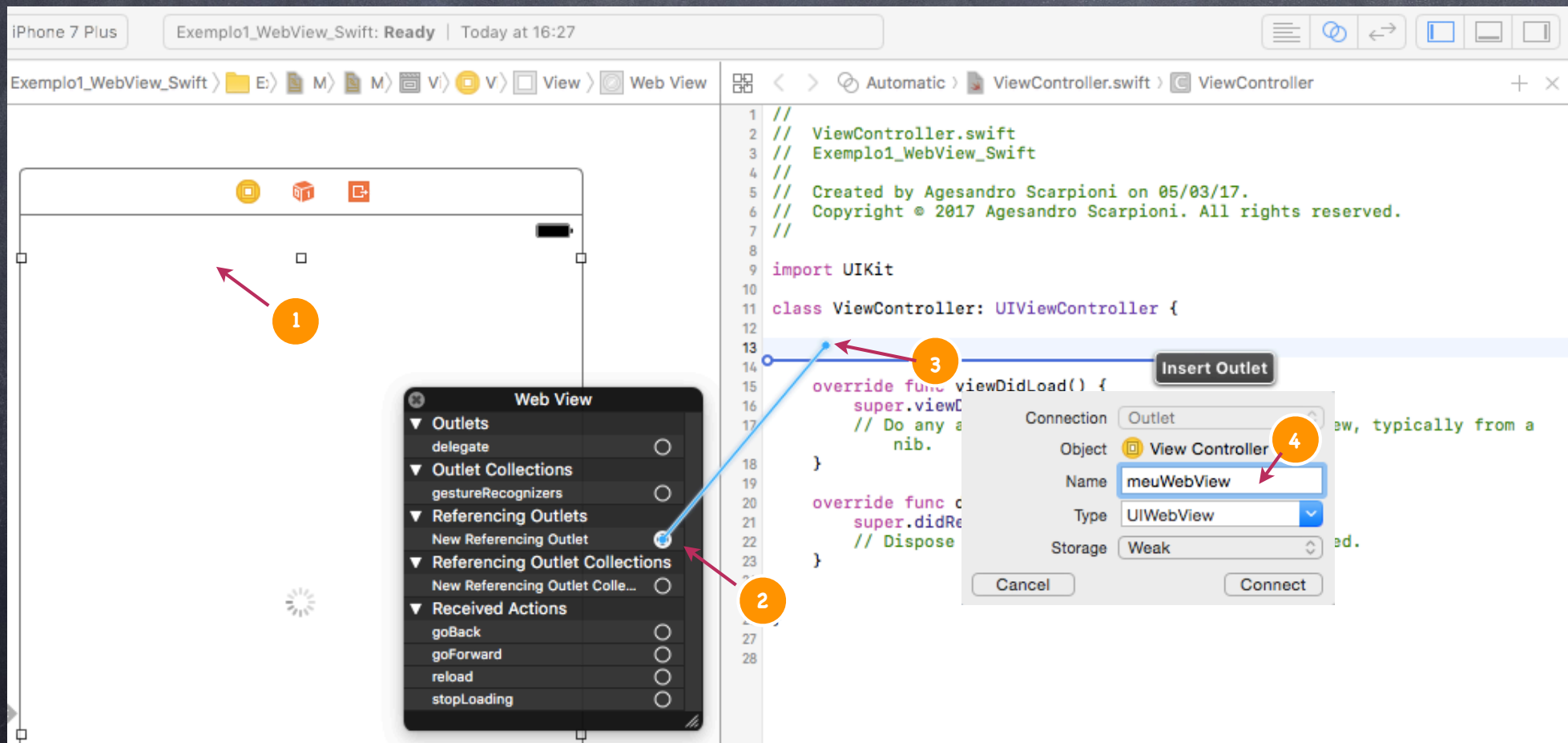
# WebView

- Clique nos pontos indicados 1 e 2 para que a View e a classe ViewController.swift fiquem simultaneamente disponíveis para que seja possível declarar os Outlets de forma automática.



# WebView

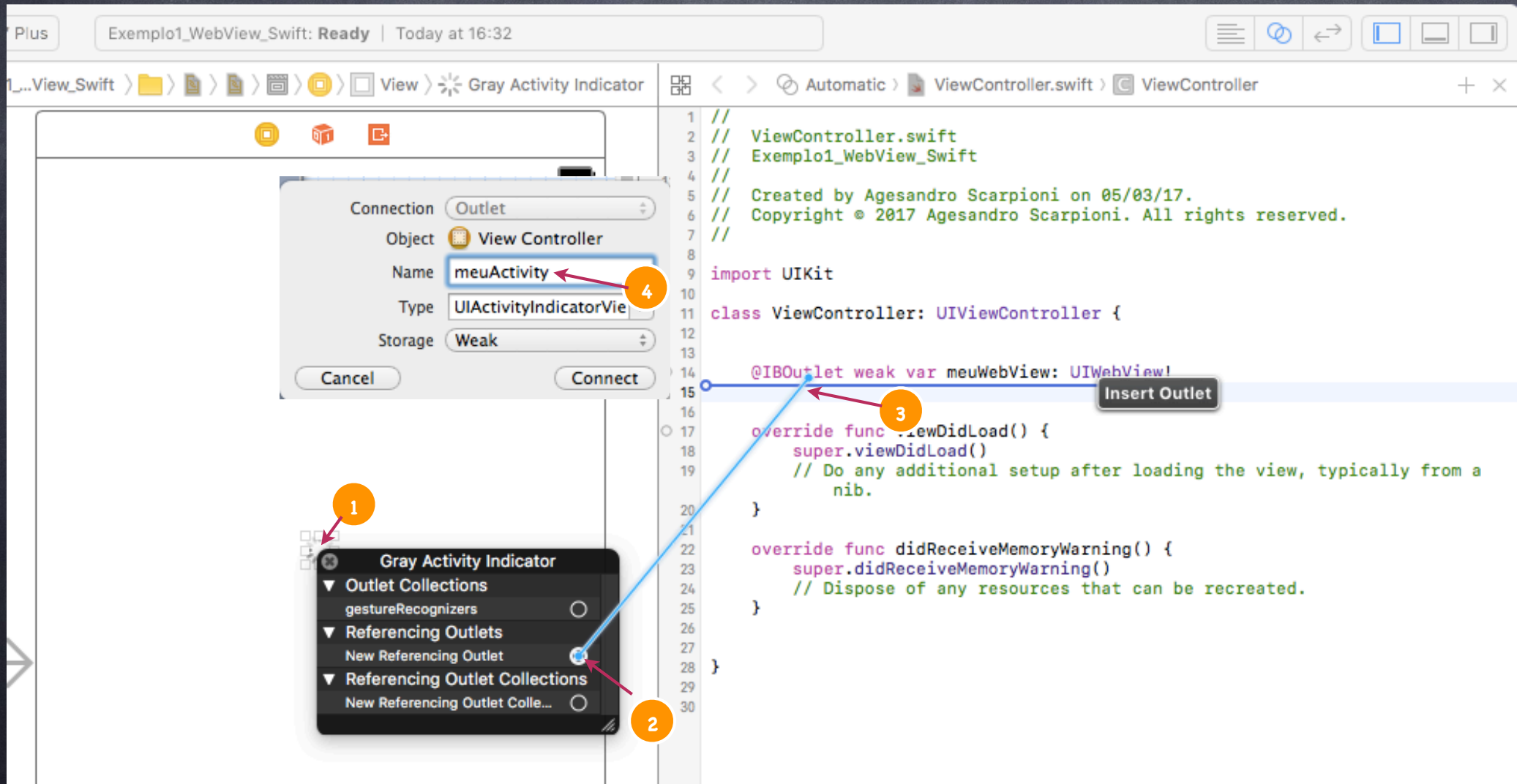
- Na classe ViewController.swift crie um outlet, clique com o botão direito sobre a UIWebView (1) e selecione New Referencing Outlet(2), arraste até a área indicada (3), ao soltar o botão do mouse irá aparecer o popup para criação do Outlet, digite em Name: meuWebView (4).





# WebView

- Clique com o botão direito no Activity(1), repita os passos para criar o IBOutlet do objeto com o nome de meuActivity(4).





# WebView

- No ViewController.swift implemente o protocolo UIWebViewDelegate (1) para que seja possível monitorar os eventos gerados pelo objeto WebView. Esse protocolo possui diversos métodos úteis, por exemplo, existem métodos para detectar se uma página já foi carregada ou se ocorreu algum erro no carregamento.

```
1 //  
2 // ViewController.swift  
3 // Exemplo1_WebView_Swift  
4 //  
5 // Created by Agesandro Scarpioni on 05/03/17.  
6 // Copyright © 2017 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 import UIKit  
10  
11 class ViewController: UIViewController, UIWebViewDelegate {  
12  
13  
14     @IBOutlet weak var meuWebView: UIWebView!  
15     @IBOutlet weak var meuActivityIndicator: UIActivityIndicatorView!  
16  
17     override func viewDidLoad() {  
18         super.viewDidLoad()  
19         // Do any additional setup after loading the view, typically  
20         // nib.  
21     }  
22  
23     override func didReceiveMemoryWarning() {  
24         super.didReceiveMemoryWarning()  
25         // Dispose of any resources that can be recreated.  
26     }  
27  
28 }  
29  
30
```



# Protocolos e Delegates

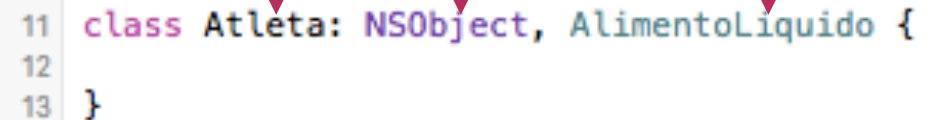
- Protocolos são templates ou interfaces de métodos e propriedades, que definem uma atividade em particular ou parte de uma funcionalidade. O protocolo não fornece a implementação dessa funcionalidade, apenas descreve como ela deve ser utilizada (assinatura do método).
- Como exemplo é possível imaginar que em duas classes uma chamada Atleta e outra Chamada Staff, necessitem de métodos para alimentação ou ingestão de líquidos, o que pode ser feito é criar um protocolo chamado AlimentoLiquido com assinaturas específicas para ingestão de líquidos como por exemplo "beberIsotonico", porém, na classe Atleta será implementado no método beberIsotonico algo como "Servir Gatorade" e para o Staff no mesmo método será implementado algo como "Servir água de Coco".
- O mesmo exemplo pode ser aplicado na classe Carro e classe Caminhão, um protocolo chamado Combustivel pode ter um método chamado "abastecer", porém, na classe Caminhão será implementado que o abastecer envie "Diesel" e para a classe Carro o abastecer envie "Etanol".



# Protocolos e Delegates

- As Classes que adotam protocolos são descritas como classes que estão em conformidade com o protocolo (conform to).
- A sintaxe que define que uma classe adota um protocolo, ou que está em conformidade (conform to) com um protocolo é: `Class Classe: SuperClasse, protocolo { }`

```
11 class Atleta: NSObject, AlimentoLiquido {  
12  
13 }
```



- Quando a classe implementar mais que um protocolo eles devem ser separados por vírgula.
- Abaixo um exemplo de um protocolo que foi criado como AlimentoLiquido e que o método beberIsotonico aguarda uma implementação na classe Atleta.

```
8  
9 import Foundation  
10  
11 protocol AlimentoLiquido: AnyClass {  
12  
13     func beberIsotonico()  
14  
15 }
```



# Protocolo UIWebViewDelegate

- O protocolo UIWebViewDelegate, possui métodos/assinaturas que podem ser adotados para se tomar ações apartir de eventos que ocorrem em um objeto WebView, como por exemplo o método didFailLoadWithError que detecta se ocorreu erro no carregamento de uma página no WebView.
- Para adotar o protocolo UIWebViewDelegate é necessário seguir 3 etapas:
  1. Adotar o protocolo
  2. Implementar o(s) método(s) desejado(s)
  3. Associar o Delegate (que pode ser feito de forma visual ou via código)

Se implementarmos o método didFailLoadWithError para exibir um alerta com a mensagem de erro informando o motivo do não carregamento de uma página sem executar a terceira etapa, nada irá acontecer, o motivo é que o WebView não tem a informação que precisa disparar o(s) método(s) do protocolo, o WebView precisa "delegar" tarefas para outra instância executar, no caso delegaremos a tarefa ao ViewController. Quando um objeto delega tarefas para outra classe implementar se diz que essa classe é o delegate do objeto, nesse exemplo o ViewController é o delegate do WebView. Delegation é um design pattern (padrão de projeto) muito utilizado nas classes de iOS.



# WebView

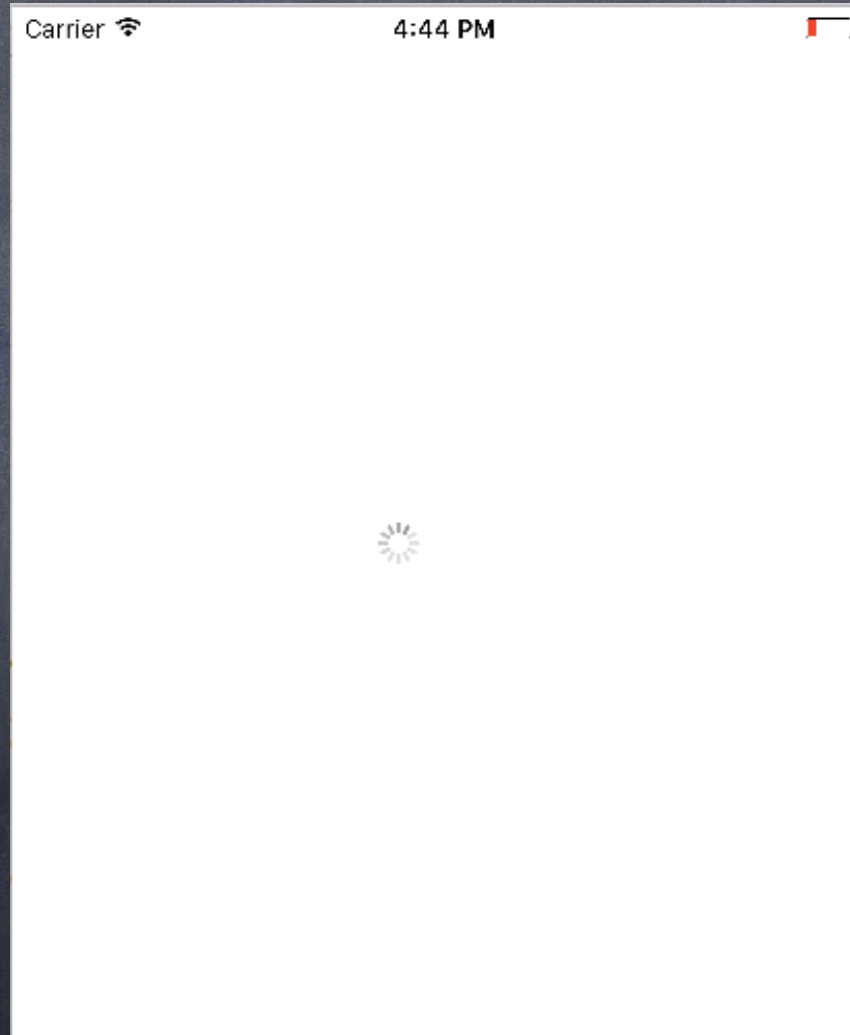
- No ViewController.swift inicialize o Activity com startAnimating (1) e declare o protocolo UIWebViewDelegate para a ViewController (2), também é possível fazer essa ligação de forma visual pelo Connections Inspector, veja página 28.

```
1 //
2 // ViewController.swift
3 // Exemplo1_WebView_Swift
4 //
5 // Created by Agesandro Scarpioni on 05/03/17.
6 // Copyright © 2017 Agesandro Scarpioni. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController, UIWebViewDelegate {
12
13
14     @IBOutlet weak var meuWebView: UIWebView!
15     @IBOutlet weak var meuActivityIndicator: UIActivityIndicatorView!
16
17     override func viewDidLoad() {
18         super.viewDidLoad()
19
20         meuActivityIndicator.startAnimating() ← 1
21
22     2 → meuWebView.delegate = self
23         //podemos fazer a linha acima conectando o delegate pelo
24         //connection inspector de forma visual, se ligarmos
25         //dessa forma, podemos comentar a linha de código acima.
26         //faremos da forma visual ao final do conjunto de slides
27     }
28
29     override func didReceiveMemoryWarning() {
30         super.didReceiveMemoryWarning()
31         // Dispose of any resources that can be recreated.
32     }
33
34
35 }
```



# WebView

- Execute o programa com command + R, veja que o Activity **não para**, pois ainda não foi implementado o delegate para que seja monitorado os eventos gerados no WebView.





# WebView

- Crie uma constante para armazenar a URL, no viewDidLoad, crie um objeto URL e um objeto URLRequest, carregue este URLRequest na WebView como é demonstrado abaixo(2).

```
1 //
2 // ViewController.swift
3 // Exemplo1_WebView_Swift
4 //
5 // Created by Agesandro Scarpioni on 05/03/17.
6 // Copyright © 2017 Agesandro Scarpioni. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController, UIWebViewDelegate {
12
13     1 → let URL_PAGINA = "http://www.vive.com/us/"
14
15     @IBOutlet weak var meuWebView: UIWebView!
16     @IBOutlet weak var meuActivityIndicator: UIActivityIndicatorView!
17
18     override func viewDidLoad() {
19         super.viewDidLoad()
20
21         meuActivityIndicator.startAnimating()
22
23         meuWebView.delegate = self
24         //podemos fazer a linha acima conectando o delegate pelo
25         //connection inspector de forma visual, se ligarmos
26         //dessa forma, podemos comentar a linha de código acima.
27         //faremos da forma visual ao final do conjunto de slides
28
29     2 → let URL_OK = URL(string: URL_PAGINA)
30     2 → let request = URLRequest(url: URL_OK!)
31     2 → meuWebView.loadRequest(request)
32     }
33 }
```

**Dica:** Podemos utilizar além do viewDidLoad o viewWillAppear a diferença que o primeiro só é executado uma vez, e o segundo é executado toda vez que a Tab é selecionada.



# WebView delegate

- Existe um método `didFailLoadWithError` que informa quando ocorreu um erro ao carregar a página, como por exemplo acesso negado, timeout, sem conexão com a internet etc, vamos implementar este método fazendo com que uma mensagem seja exibida com o erro caso ocorra. Isto só foi possível porque em `viewController.swift` declaramos o protocolo `UIWebViewDelegate`

```
39  
40 func webView(_ webView: UIWebView, didFailLoadWithError error: Error) {  
41     var msg:String  
42     msg = "Erro: \(error.localizedDescription)"  
43  
44     let alerta = UIAlertController(  
45         title: "Aviso",  
46         message: msg,  
47         preferredStyle: UIAlertControllerStyle.alert)  
48  
49     alerta.addAction(UIAlertAction(  
50         title: "OK",  
51         style: UIAlertActionStyle.default,  
52         handler: nil))  
53  
54     present(alerta, animated: true, completion: nil)  
55 }
```

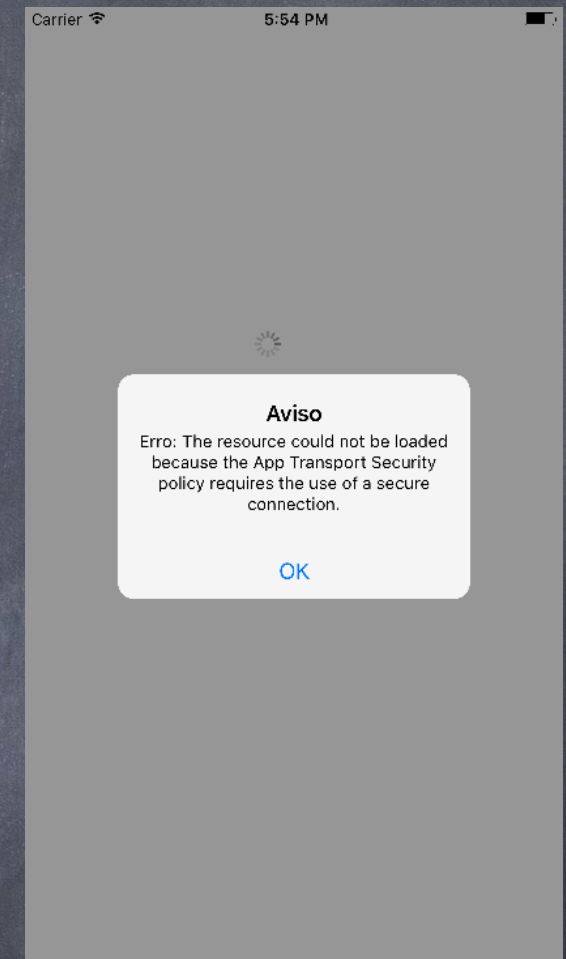
- Sem associar o Delegate (linha do delegate no `didLoad`) essa parte não funciona, ou seja, caso tenha um erro no endereço da URL o programa não irá mostrar o alerta, com a linha digitada no `didLoad` o método passa a funcionar, lembre-se que é possível associar o protocolo `UIWebViewDelegate` visualmente pelo Connections Inspector.



# ATS

- Ao executar o App o erro abaixo será exibido, para o iOS9 a Apple tomou uma decisão radical desativando todo o tráfego HTTP não seguro a partir de aplicativos iOS, como parte do ATS (App Transport Security).
- Você pode desativar ATS no arquivo info.plist adicionando as seguintes linhas:

```
<key>NSAppTransportSecurity</key>
<dict>
  <key>NSAllowsArbitraryLoads</key>
  <true/>
</dict>
```
- Se seu aplicativo não tem uma boa razão para desativar ATS, existe o risco de rejeição. Definir NSAllowsArbitraryLoads como true, irá permitir que o App funcione, porém, a Apple foi clara em rejeitar aplicativos que usam esse sinalizador sem uma razão específica. A principal razão para usar esse sinalizador seria para compartilhamento de link, navegador Web personalizado etc, mesmo assim, a Apple ainda espera que seja incluída exceções que impõem a ATS para as URLs que estão no seu controle.



OBS: Cuidado com a solução NSAllowsArbitraryLoads. Não é a correção recomendada da Apple



- Você pode adicionar exceções para controlar domínios específicos(1) em seu info.plist.

```
<key>NSAppTransportSecurity</key>
<dict>
  <key>NSExceptionDomains</key>
  <dict>
    1 → <key>testdomain.com</key>
    <dict>
      <key>NSIncludesSubdomains</key>
      <true/>
      <key>NSExceptionAllowsInsecureHTTPLoads</key>
      <true/>
      <key>NSExceptionRequiresForwardSecrecy</key>
      <true/>
      <key>NSExceptionMinimumTLSVersion</key>
      <string>TLSv1.2</string>
      <key>NSThirdPartyExceptionAllowsInsecureHTTPLoads</key>
      <false/>
      <key>NSThirdPartyExceptionRequiresForwardSecrecy</key>
      <true/>
      <key>NSThirdPartyExceptionMinimumTLSVersion</key>
      <string>TLSv1.2</string>
      <key>NSRequiresCertificateTransparency</key>
      <false/>
    </dict>
  </dict>
</dict>
```

Se você precisa de acesso a URLs específicos, você precisa escrever exceções para esses domínios, não use `NSAllowsArbitraryLoads` setada como `True`. Você pode encontrar mais informações [WWDC 2015 session 711 NSURLSession clicando aqui](#).

- Todas as chaves para `NSExceptionDomains` são opcionais. No link do vídeo do [WWDC 2015 session 703, "Privacy and Your App"](#), 30:19 o apresentador não entrou em detalhes sobre qualquer uma das chaves.
- Para todos os detalhes de `NSAppTransportSecurity` veja no site da Apple o [info.plist reference clicando aqui](#).



# Info.plist

- Existem 2 formas de alterar o arquivo info.plist, escolha a forma que lhe agrada nos próximos slides.



# Info.plist (forma 1)

- Abra o arquivo info.plist(1), clique no símbolo + (2), role as opções para encontrar: App Transport Security Settings (3), clique no símbolo do triângulo indicando-o para baixo (4), dessa forma é possível incluir um sub item clicando no + (5), role as opções para encontrar: Allow Arbitrary Loads (6), selecione YES (7).

The image shows a sequence of four screenshots from the Xcode IDE, illustrating the steps to configure App Transport Security Settings in the Info.plist file. The steps are numbered 1 through 7.

**Step 1:** The left sidebar shows the project structure. The file **Info.plist** is selected and highlighted with a red arrow and a yellow circle labeled 1.

**Step 2:** The right pane shows the **Key** and **Type** columns of the Info.plist. The **Application Category** key is selected, and a red arrow points to the plus sign (+) icon next to it, labeled with a yellow circle 2.

**Step 3:** A context menu is shown over the **Application Category** key. The option **App Transport Security Settings** is selected, indicated by a red arrow and a yellow circle 3.

**Step 4:** The **App Transport Security Settings** key is now visible in the list, and a red arrow points to the downward-pointing triangle icon next to it, labeled with a yellow circle 4.

**Step 5:** A plus sign (+) icon is clicked next to the **App Transport Security Settings** key to add a new item, indicated by a red arrow and a yellow circle 5.

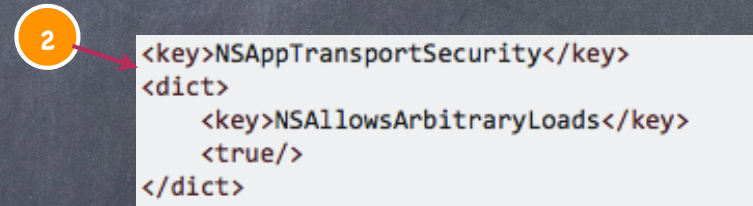
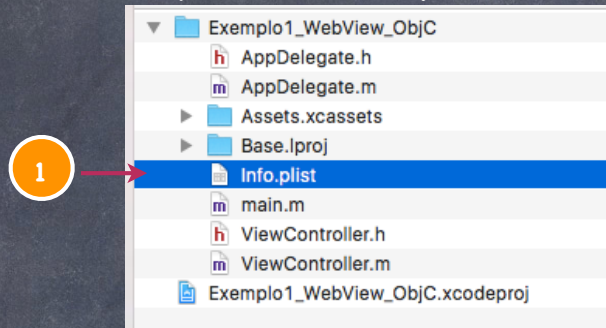
**Step 6:** The new item **Allow Arbitrary Loads** is added to the list under **App Transport Security Settings**, indicated by a red arrow and a yellow circle 6.

**Step 7:** The value for **Allow Arbitrary Loads** is set to **YES**, indicated by a red arrow and a yellow circle 7.

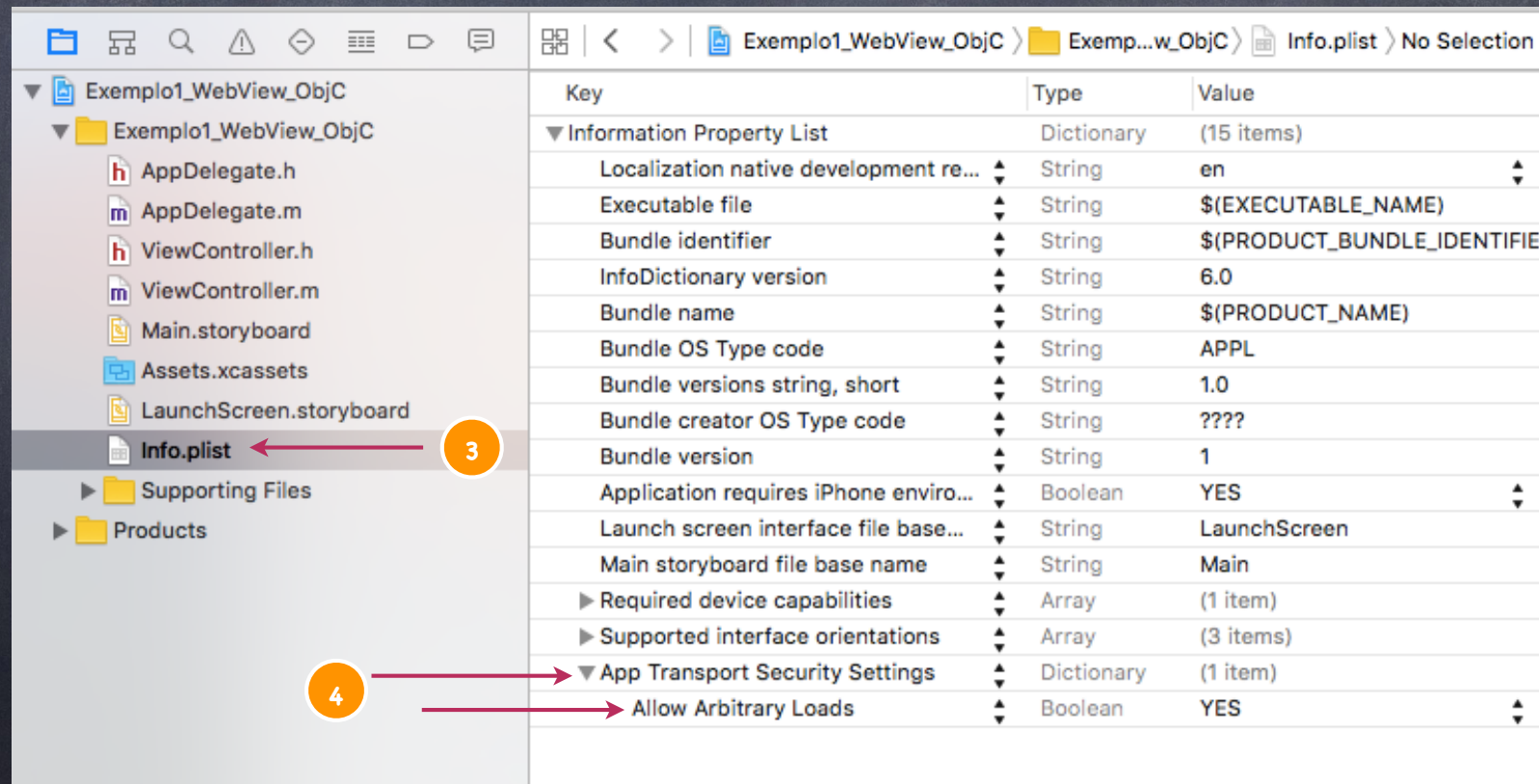


# Info.plist (forma 2)

- Abra no Finder (1) o arquivo info.plist com um editor de texto, copiar o trecho (2) e fechar, no Xcode clique no info.plist (3) para aparecer a linha do ATS desativado (4).



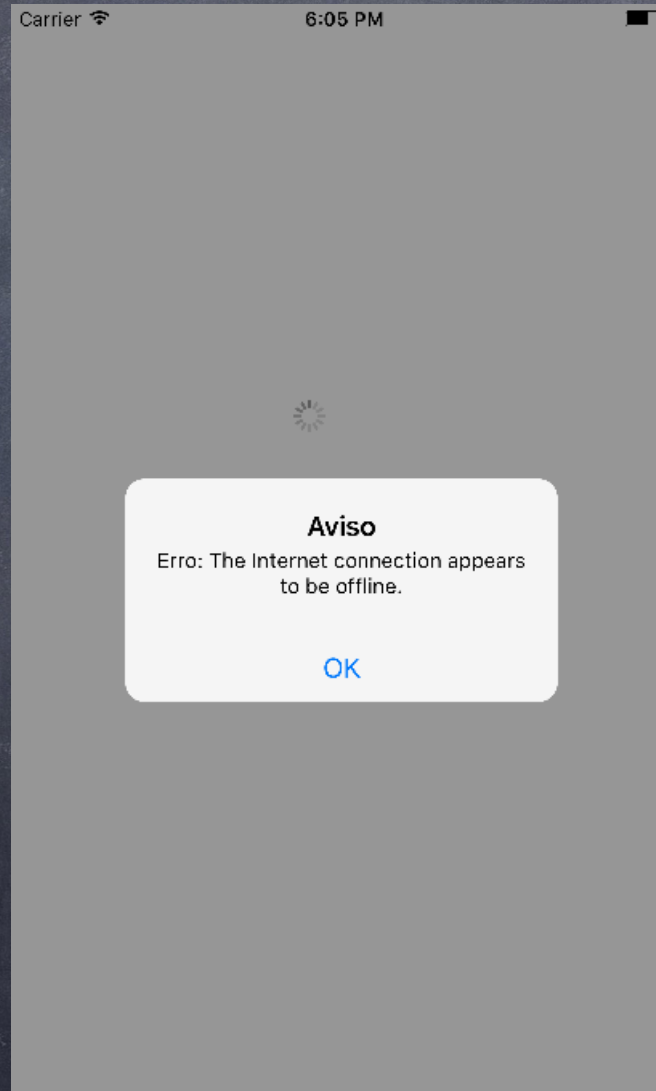
Dica: A forma 2 é ideal quando você possui o sinalizador no formato de tag < >





# WebView delegate

- Command + R, ao executar sem a conexão com a internet o erro abaixo será exibido, porém, ainda é necessário parar a animação do activity, quando a página for carregada.





# WebView delegate

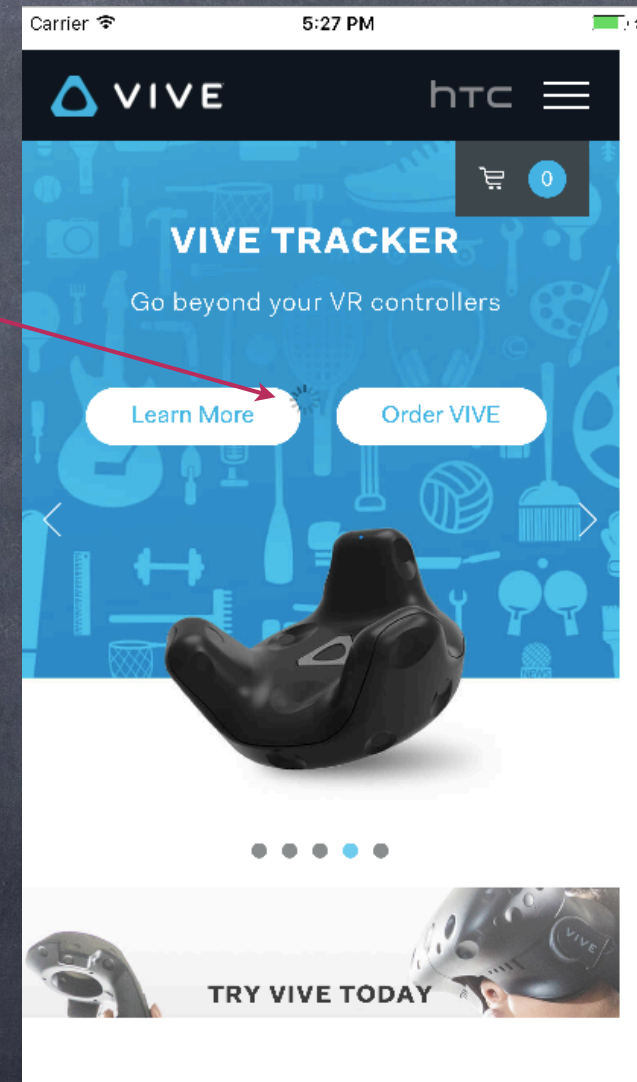
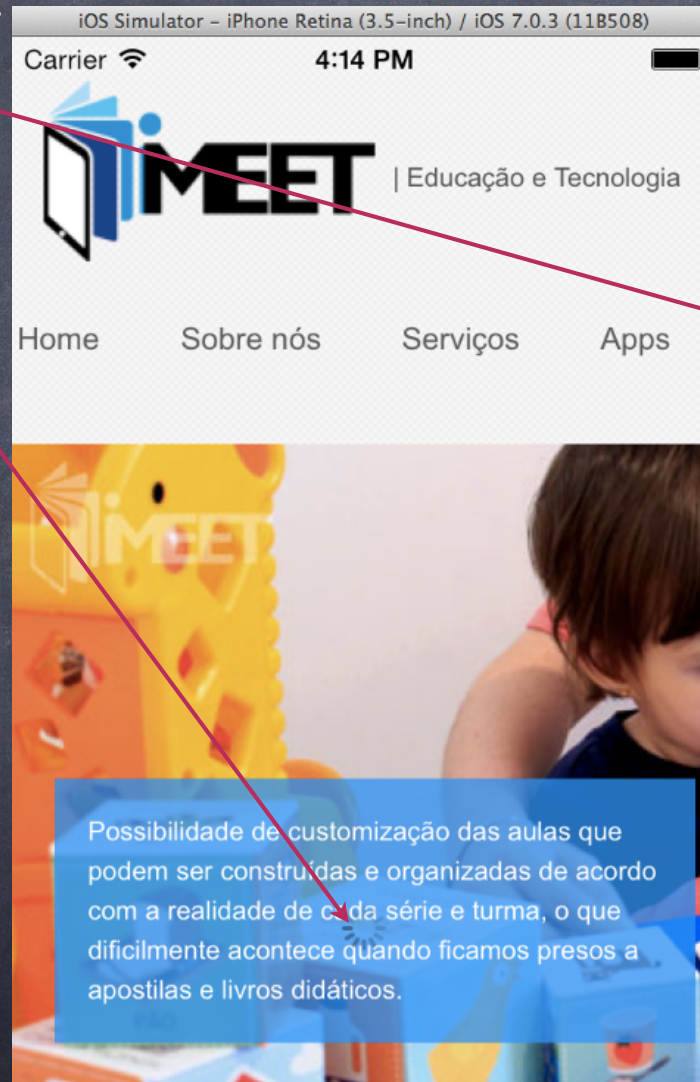
- O método `webViewDidFinishLoad` do protocolo `UIWebViewDelegate` é chamado no final da requisição, quando uma página é carregada com sucesso.

```
56  
57     func webViewDidFinishLoad(_ webView: UIWebView) {  
58         meuActivity.stopAnimating()  
59     }  
60
```



# WebView delegate

- Quando for executado o simulador com a internet ligada irá aparecer a página e o activity irá parar.





# Esconder o Activity

- Existe duas formas de esconder o Activity

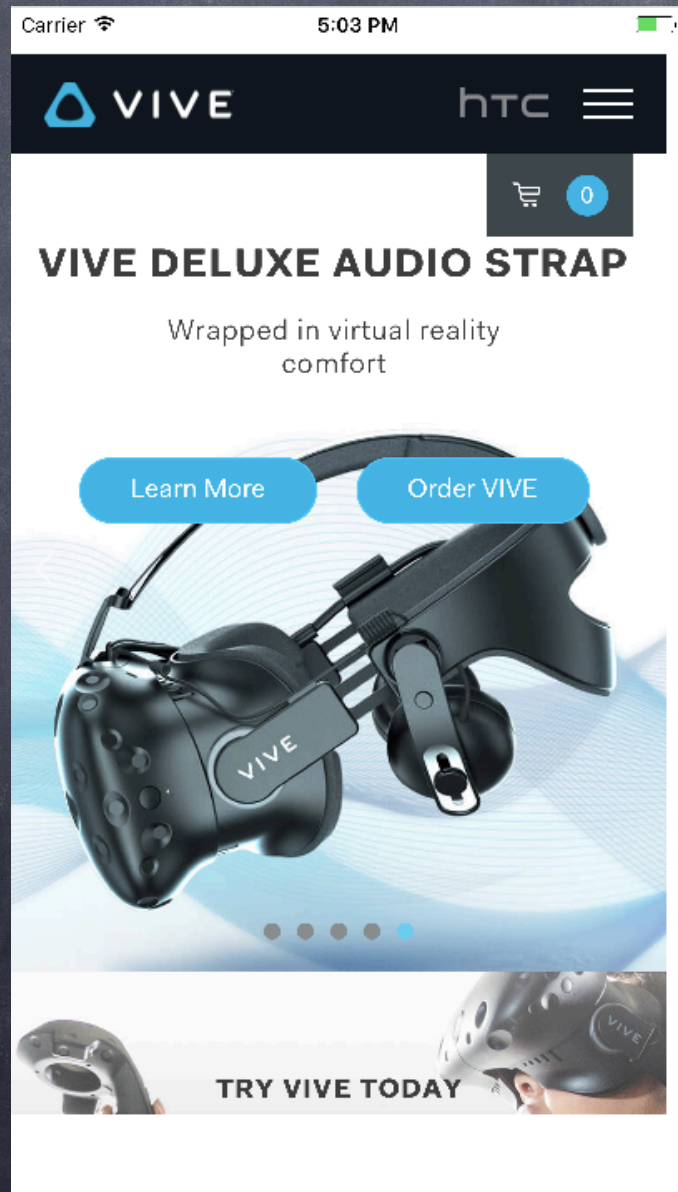
```
59     func webViewDidFinishLoad(_ webView: UIWebView) {  
60         meuActivity.stopAnimating()  
61         1 → meuActivity.isHidden = true  
62     }
```

```
18     override func viewDidLoad() {  
19         super.viewDidLoad()  
20  
21         meuActivity.startAnimating()  
22  
23         meuWebView.delegate = self  
24         //podemos fazer a linha acima conectando o delegate pelo  
25         //connection inspector de forma visual, se ligarmos  
26         //dessa forma, podemos comentar a linha de código acima.  
27         //faremos da forma visual ao final do conjunto de slides  
28  
29         let URL_OK = URL(string: URL_PAGINA)  
30         let request = URLRequest(url: URL_OK!)  
31         meuWebView.loadRequest(request)  
32         2 → meuActivity.hidesWhenStopped = true //outra forma de esconder o activity  
33         //não precisa da chamada do hidden no webViewDidFinishLoad  
34     }
```



# WebView delegate

- Quando for executado o simulador com a internet ligada irá aparecer a página e o activity irá desaparecer.





# WebView

- Para implementar o protocolo `UIWebViewDelegate` visualmente selecione o `WebView` (1), em `connections inspector` (2), ligue o delegate ao `ViewController` arrastando do radio button (3) até o `ViewController` (4). Depois dessa ligação a linha `"meuWebView.delegate=self"` no `viewDidLoad` pode ser comentada, ou seja, existem duas formas de associar o protocolo, pelo código ou de forma visual pelo `connections inspector`.

