WebView

Utilizando StoryBoard

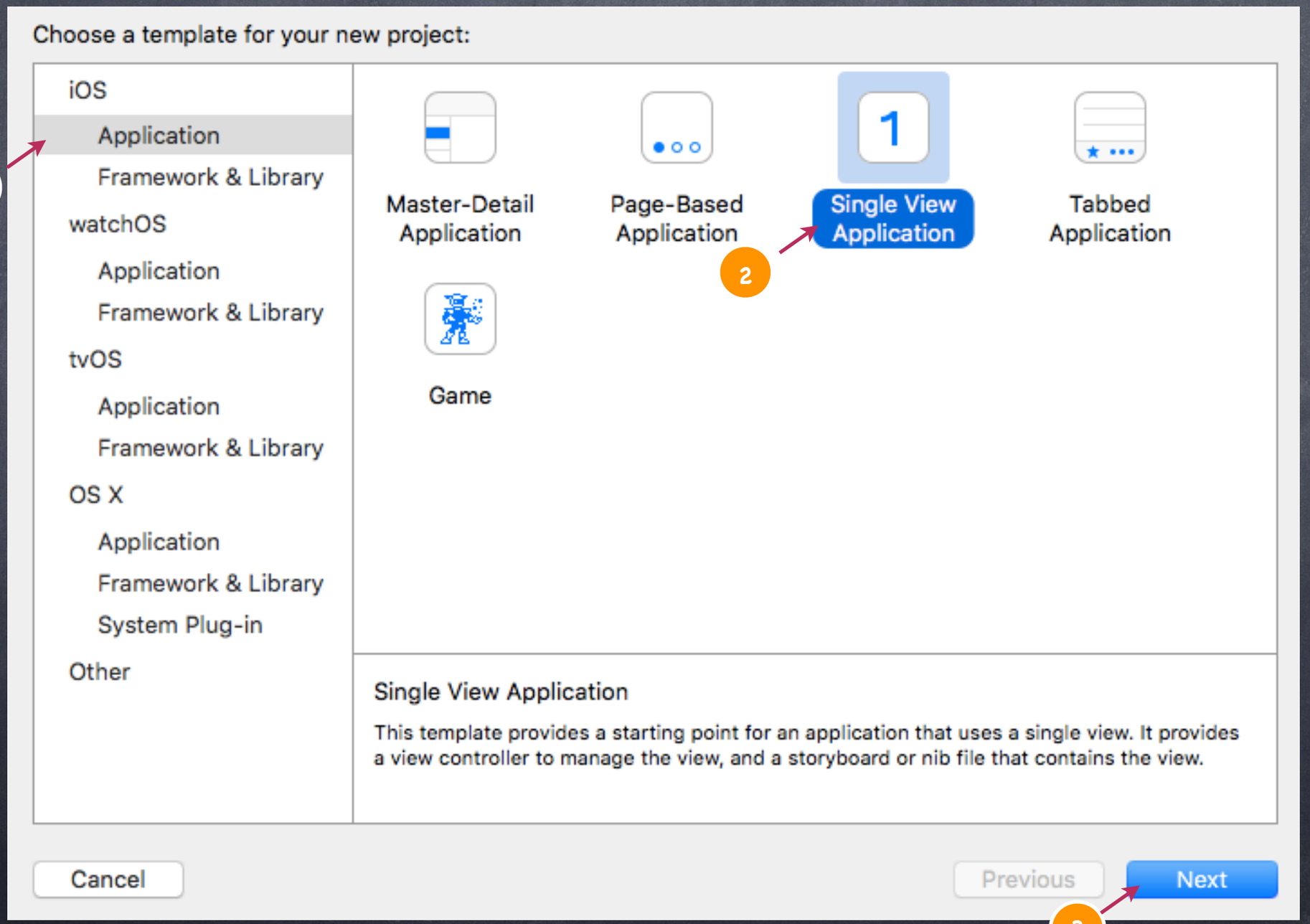
X-Code com Swift
Prof. Agesandro Scarpioni

WebView

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

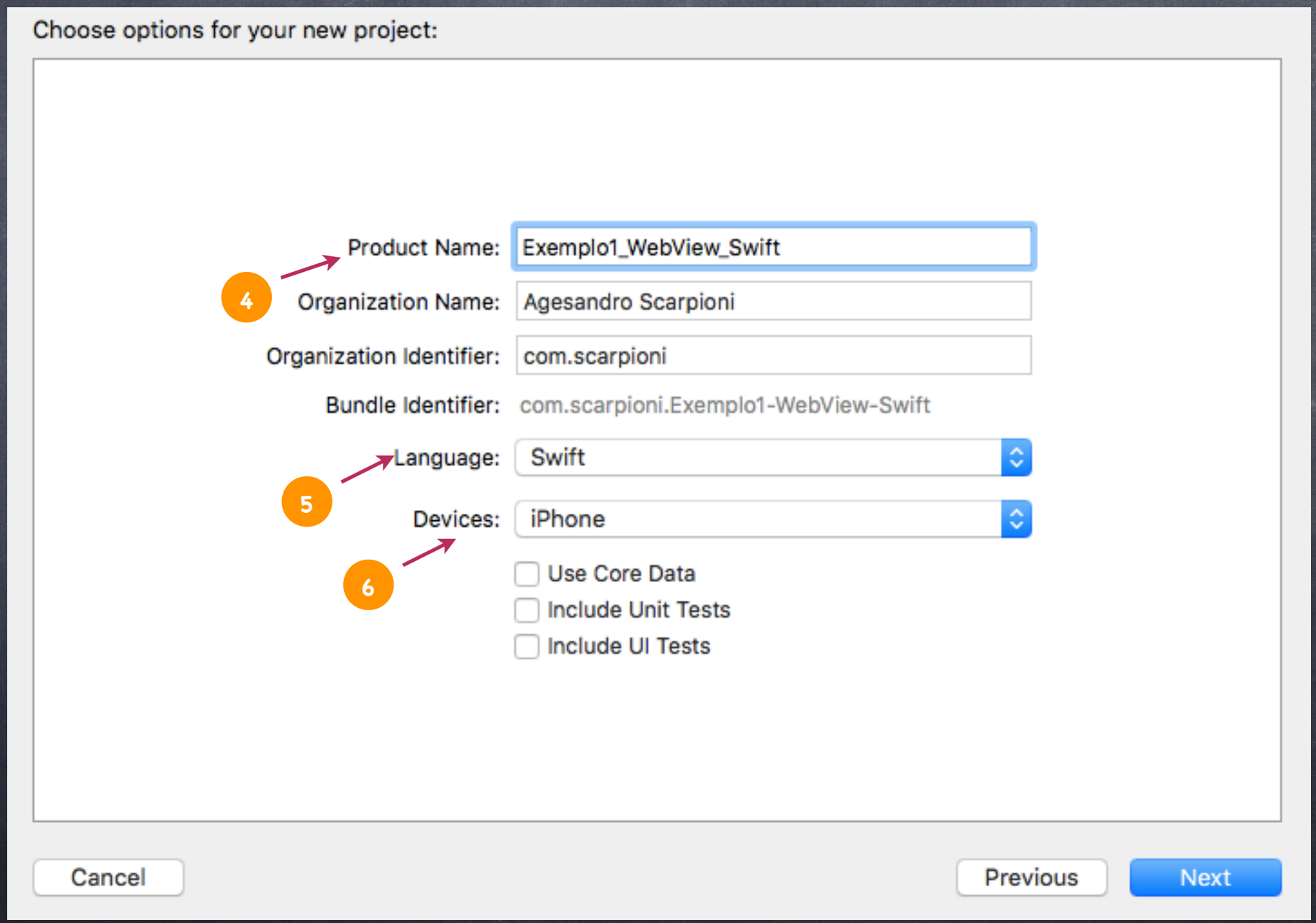
WebView

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



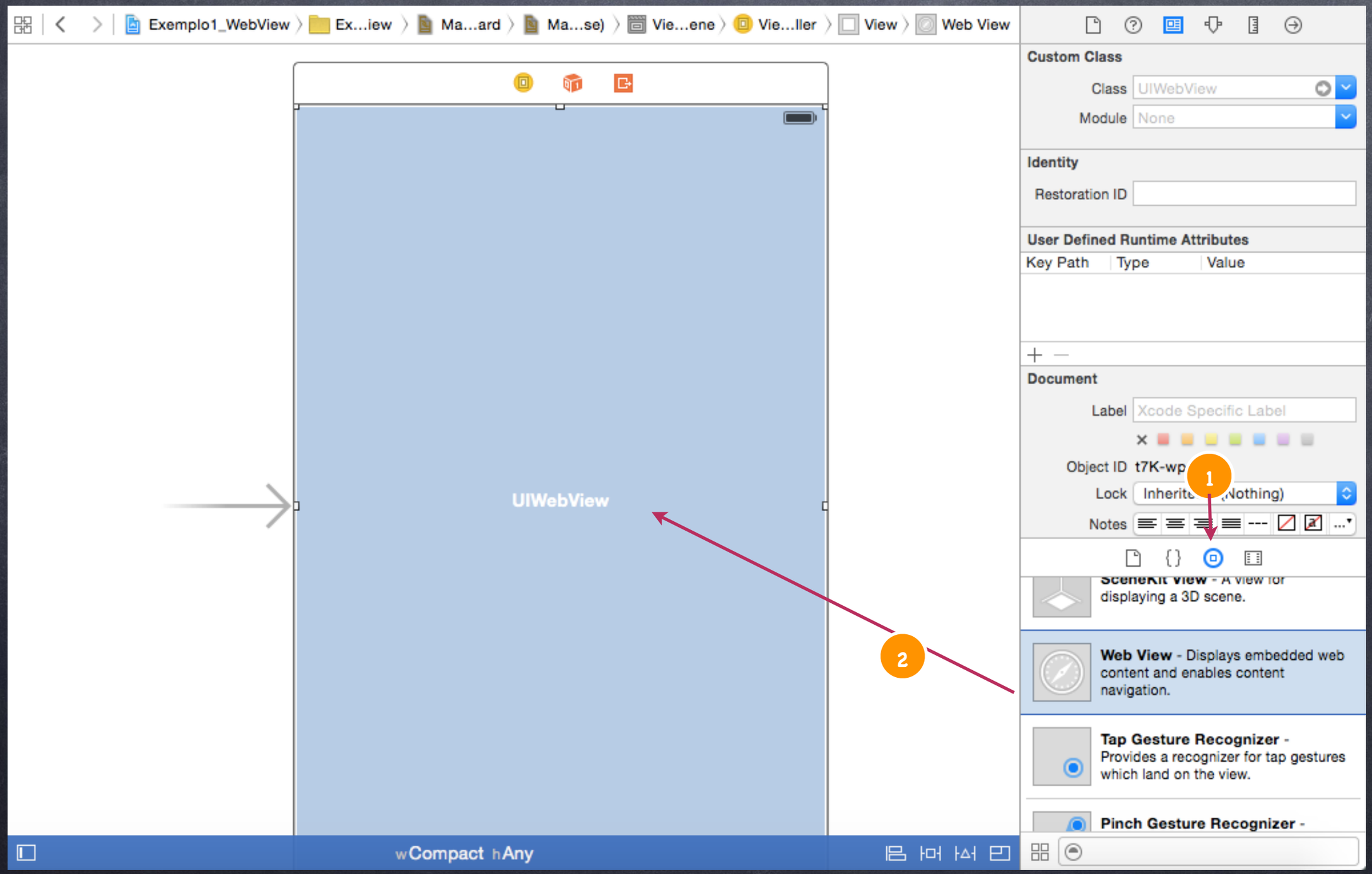
WebView

- Nomeie o projeto como: "Exemplo1_WebView_Swift" (4), selecione a linguagem Swift(5) e o device iPhone(6).



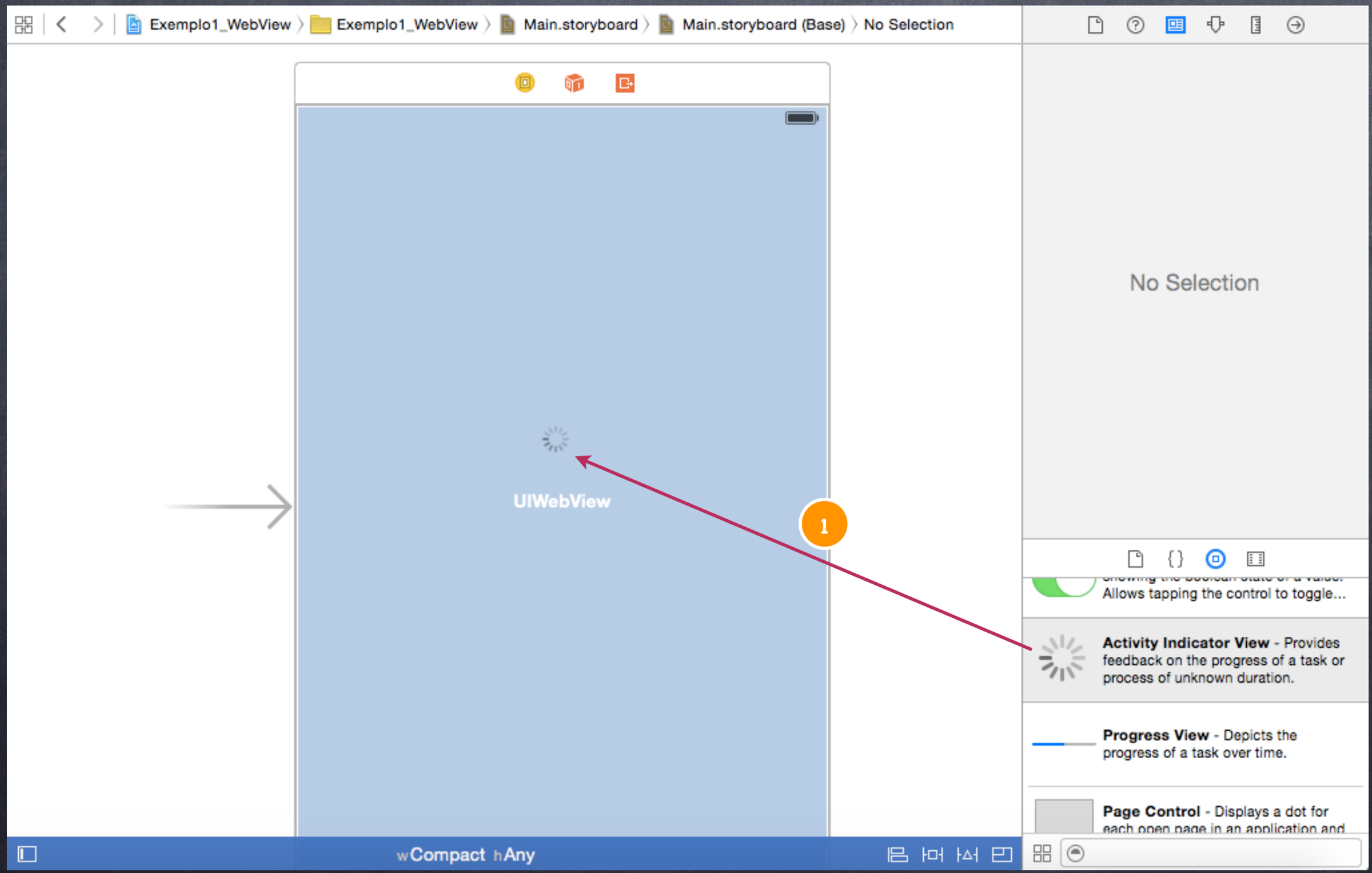
WebView

- Insira um objeto WebView em nossa View, basta arrastá-lo.



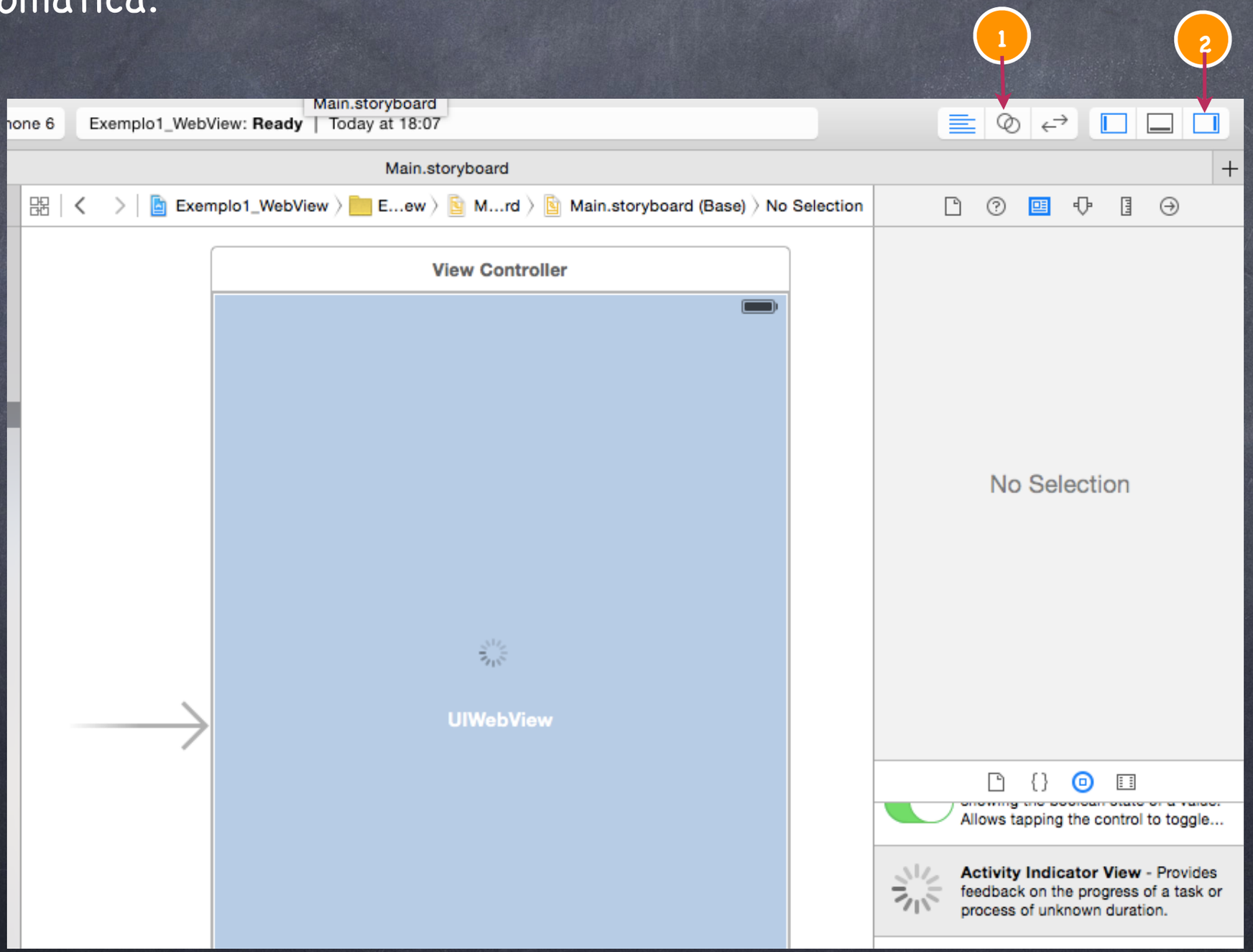
WebView

- Insira um objeto Activity Indicator View (1) junto de sua WebView



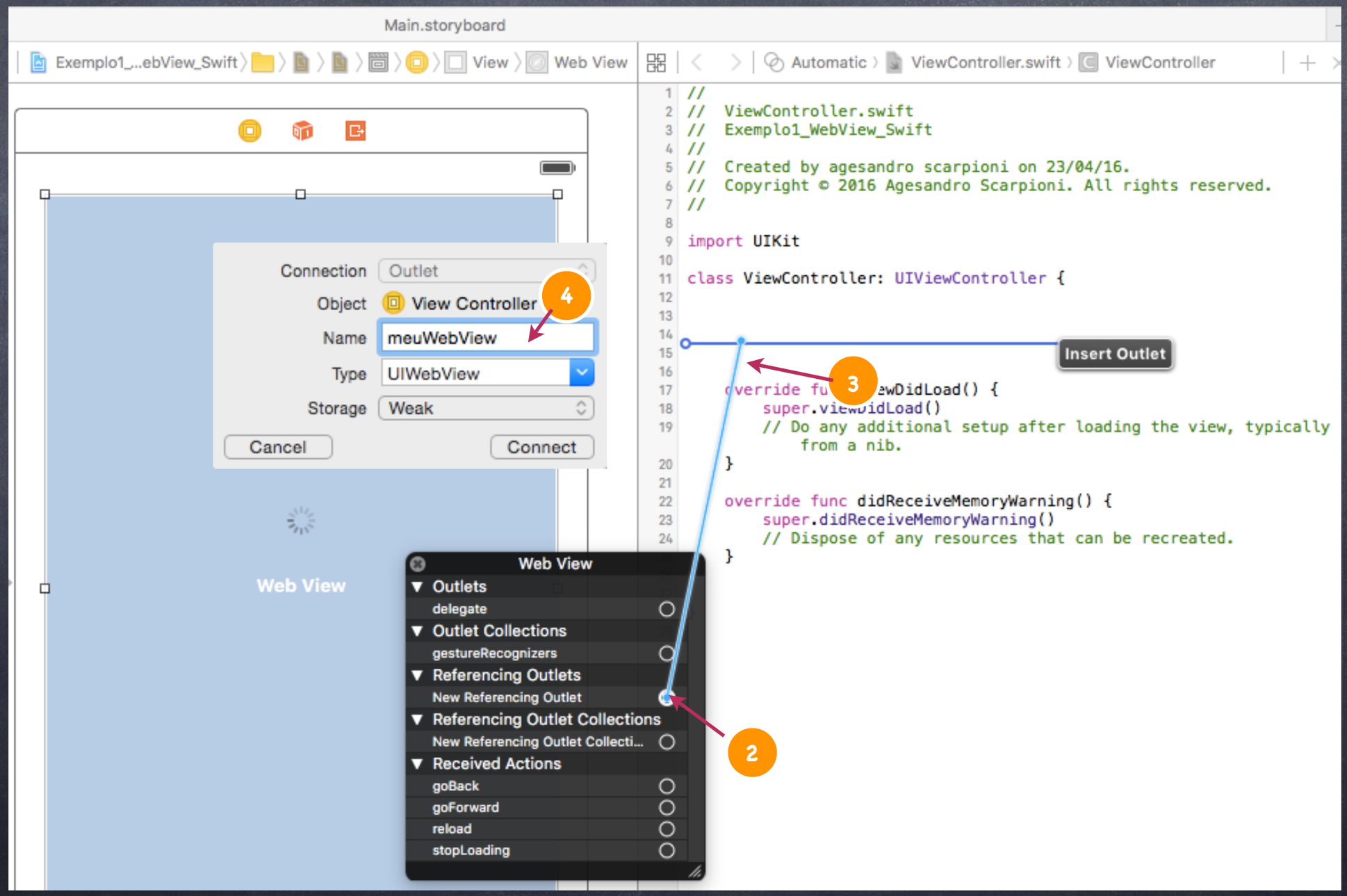
WebView

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



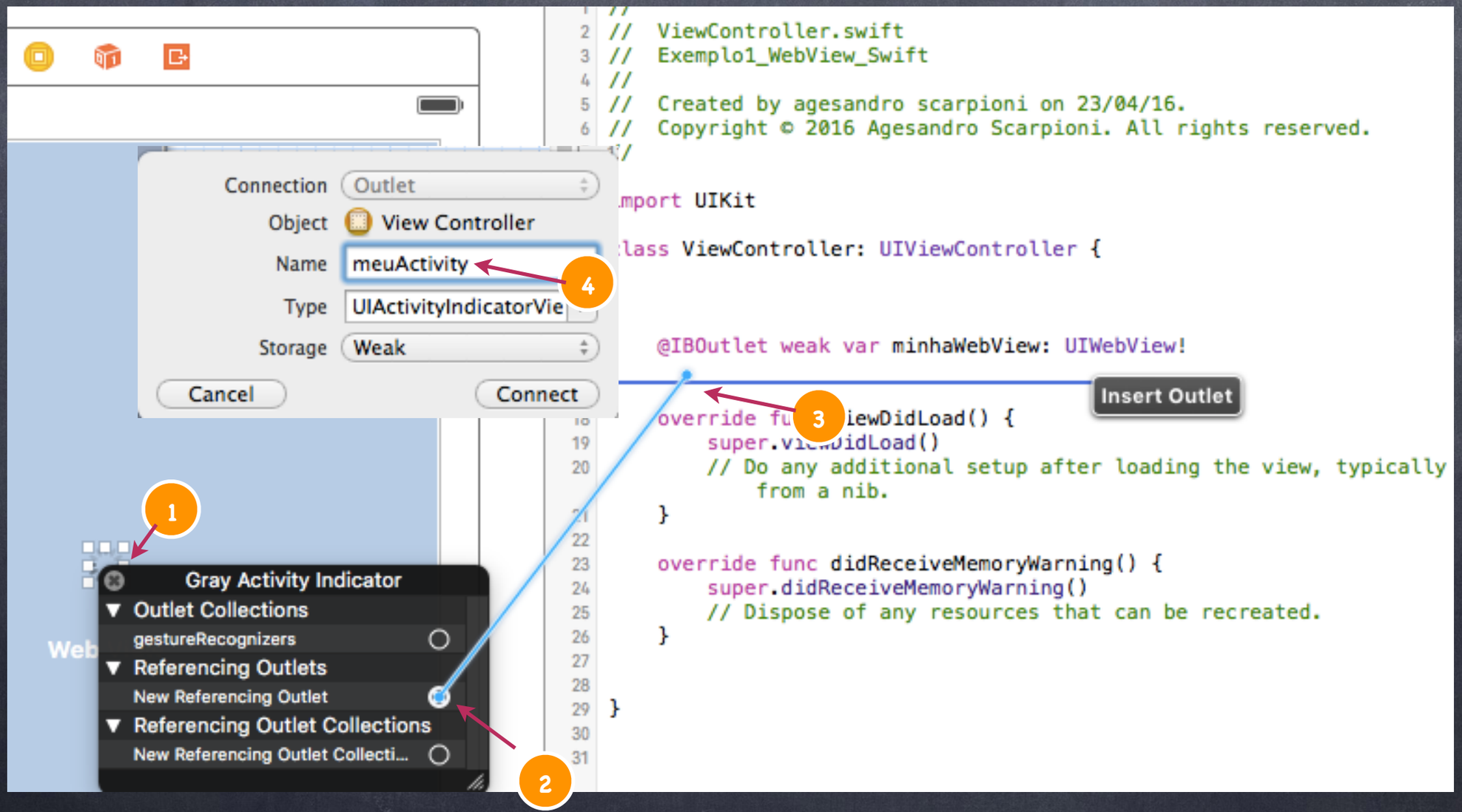
WebView

- Na classe ViewController.swift crie um outlet, clique com o botão direito sobre a UIWebView 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, 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 23/04/16.
6  //  Copyright © 2016 Agesandro Scarpioni. All rights reserved.
7  //
8
9  import UIKit
10
11 class ViewController: UIViewController, UIWebViewDelegate {
12
13
14
15     @IBOutlet weak var minhaWebView: UIWebView!
16
17     @IBOutlet weak var minhaActivity: UIActivityIndicatorView!
18
```

1

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.

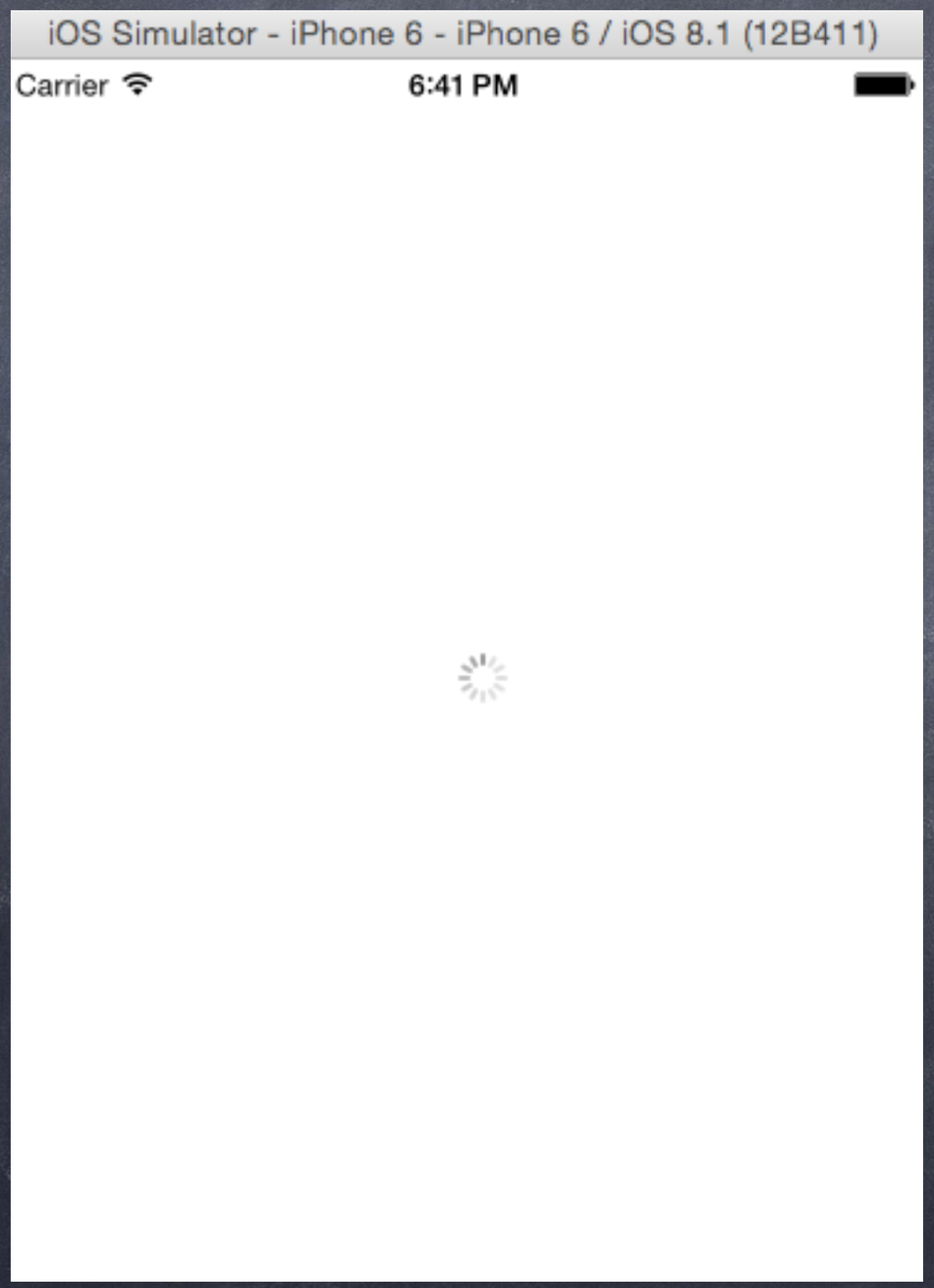
```swift
//
//  ViewController.swift
//  Exemplo1_WebView_Swift
//
//  Created by agesandro scarpioni on 23/04/16.
//  Copyright © 2016 Agesandro Scarpioni. All rights reserved.
//

import UIKit

class ViewController: UIViewController, UIWebViewDelegate {

    @IBOutlet weak var meuWebView: UIWebView!

    @IBOutlet weak var meuActivity: UIActivityIndicatorView!

    override func viewDidLoad() {
        super.viewDidLoad()

        meuActivity.startAnimating()

        meuWebView.delegate = self
        //podemos fazer a linha acima conectando o delegate
        //pelo connection inspector de forma visual, se ligarmos
        //dessa forma podemos comentar a linha de código acima.
        //Faremos isso ao final do slide para mostrar como fazê-lo
        //de forma visual

    }
```

1 ← `meuActivity.startAnimating()`

2 → `meuWebView.delegate = self`

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 demostrado abaixo(2).

```
8
9  import UIKit
10
11 class ViewController: UIViewController, UIWebViewDelegate{
12
13     let URL_PAGINA = "http://www.imeet.com.br/index.html"
14
15     @IBOutlet weak var meuWebView: UIWebView!
16
17     @IBOutlet weak var meuActivity: UIActivityIndicatorView!
18
19     override func viewDidLoad() {
20         super.viewDidLoad()
21
22         meuActivity.startAnimating()
23
24         meuWebView.delegate = self
25         //podemos fazer a linha acima conectando o delegate
26         //pelo connection inspector de forma visual, se ligarmos
27         //dessa forma podemos comentar a linha d código acima.
28         //Faremos isso ao final do slide para msotrar como fazê-lo
29         //de forma visual
30
31         let url = NSURL(string: URL_PAGINA )
32         let request = NSURLRequest(URL: url!)
33         meuWebView.loadRequest(request)
34     }
```

1

2

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

WebView delegate

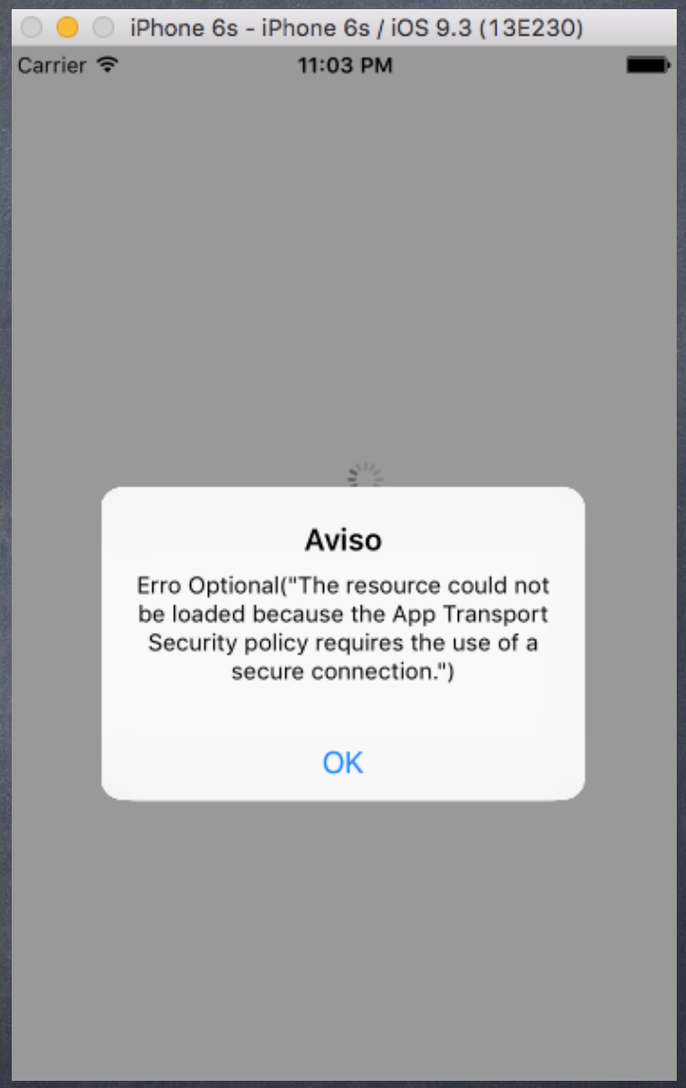
- Existe um método didFailLoadWithError que informa quando ocorreu um erro ao carregar a página, como por exemplo acesso negado, timeout, sem conexão com a internet etc, vamos implementar este método fazendo com que uma mensagem seja exibida com o erro caso ocorra. Isto só foi possível porque em viewController.swift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

```
41      func webView(webView: UIWebView, didFailLoadWithError error: NSError?) {
42
43          var msg: String
44          msg = "Erro:  \(error!.localizedDescription)"
45
46          let alerta = UIAlertController(
47              title: "Aviso",
48              message: msg,
49              preferredStyle: UIAlertControllerStyle.Alert)
50
51          alerta.addAction(UIAlertAction(
52              title: "OK",
53              style: UIAlertActionStyle.Default,
54              handler: nil))
55
56          presentViewController(alerta, animated: true, completion: nil)
57      }
```

- 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 você está no seu controle.

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

ATS

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

```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSExceptionDomains</key>
    <dict>
        <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>
```

1

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 NSURLSesssion 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.

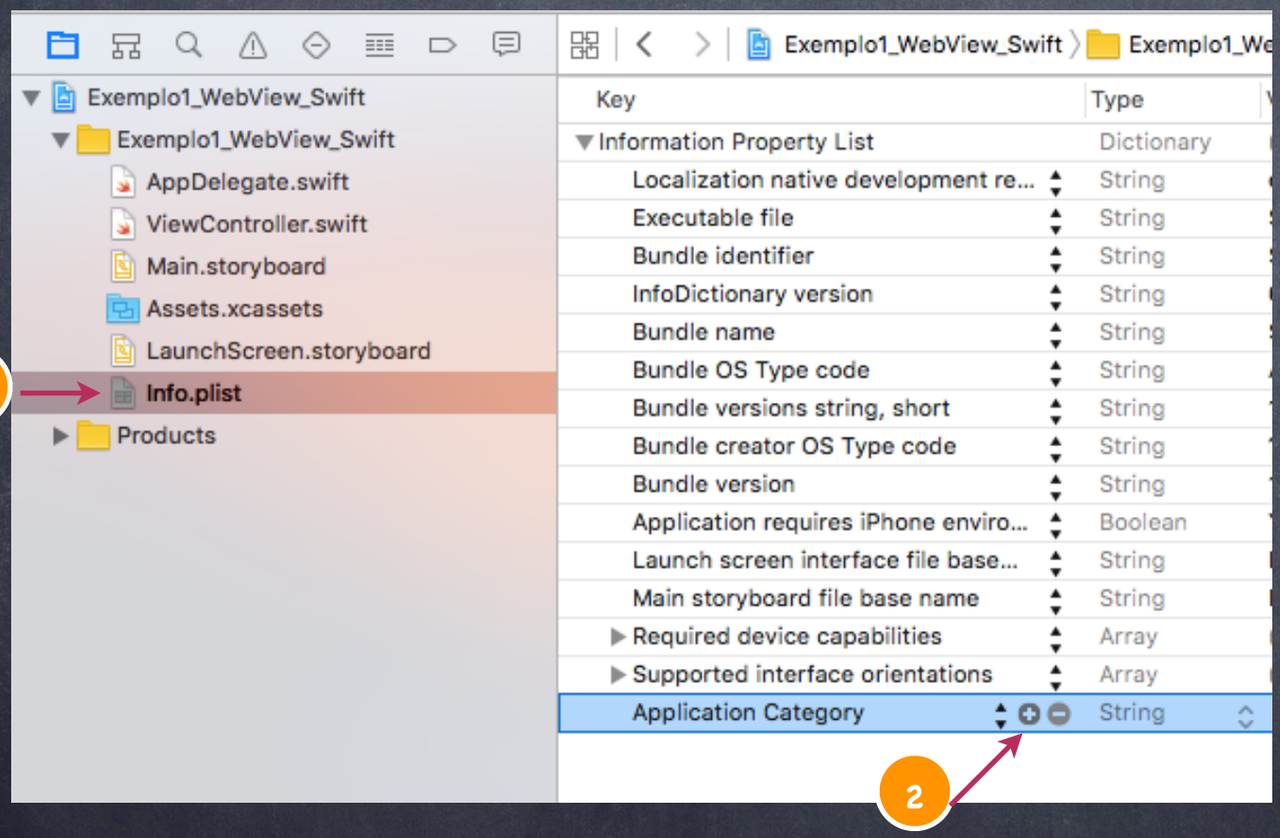
- Para todos os detalhes de NSAppTransportSecurity veja no site da Apple o info.plist reference clicando aqui.

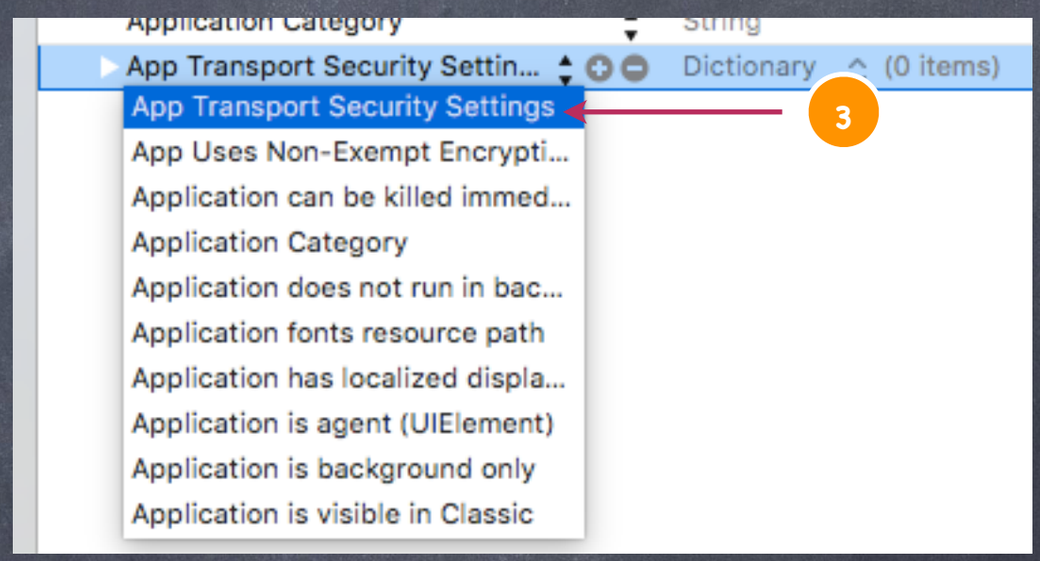
Info.plist

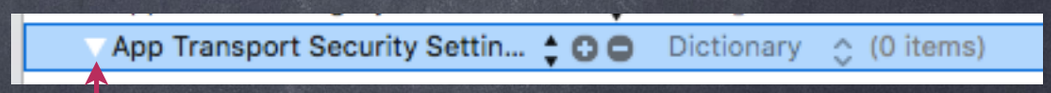
- Existem 2 formas de alterar o arquivo info.plist, escolha a forma que lhe agradar nos próximos slides.

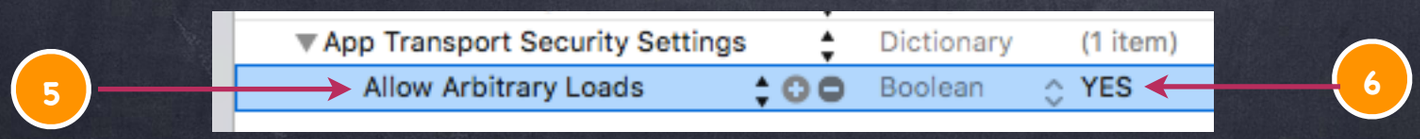
Info.plist (forma 1)

- 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, role as opções para encontrar: Allow Arbitrary Loas (5), selecione YES (6).



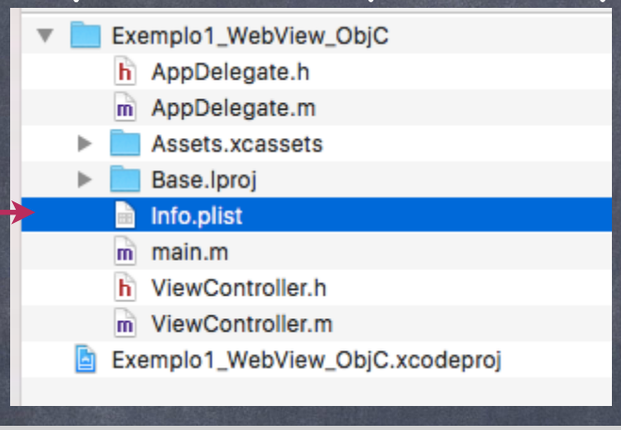






Info.plist (forma 2)

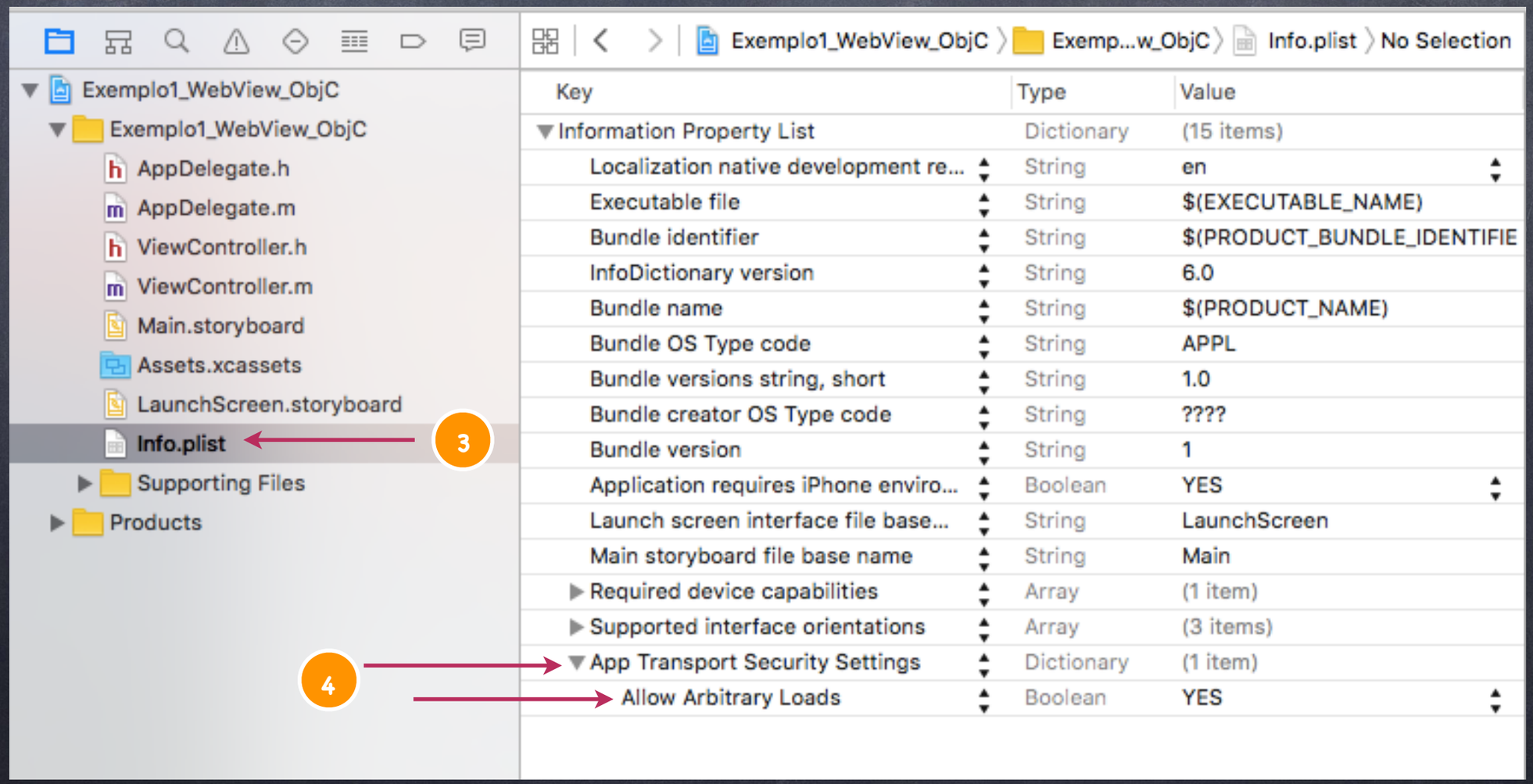
- Abra no Finder (1) o arquivo info.plist com um editor de texto, copiar o trecho (2) e fechar, no Xcode clique no info.plist (3) para aparecer a linha do ATS desativado (4).



2

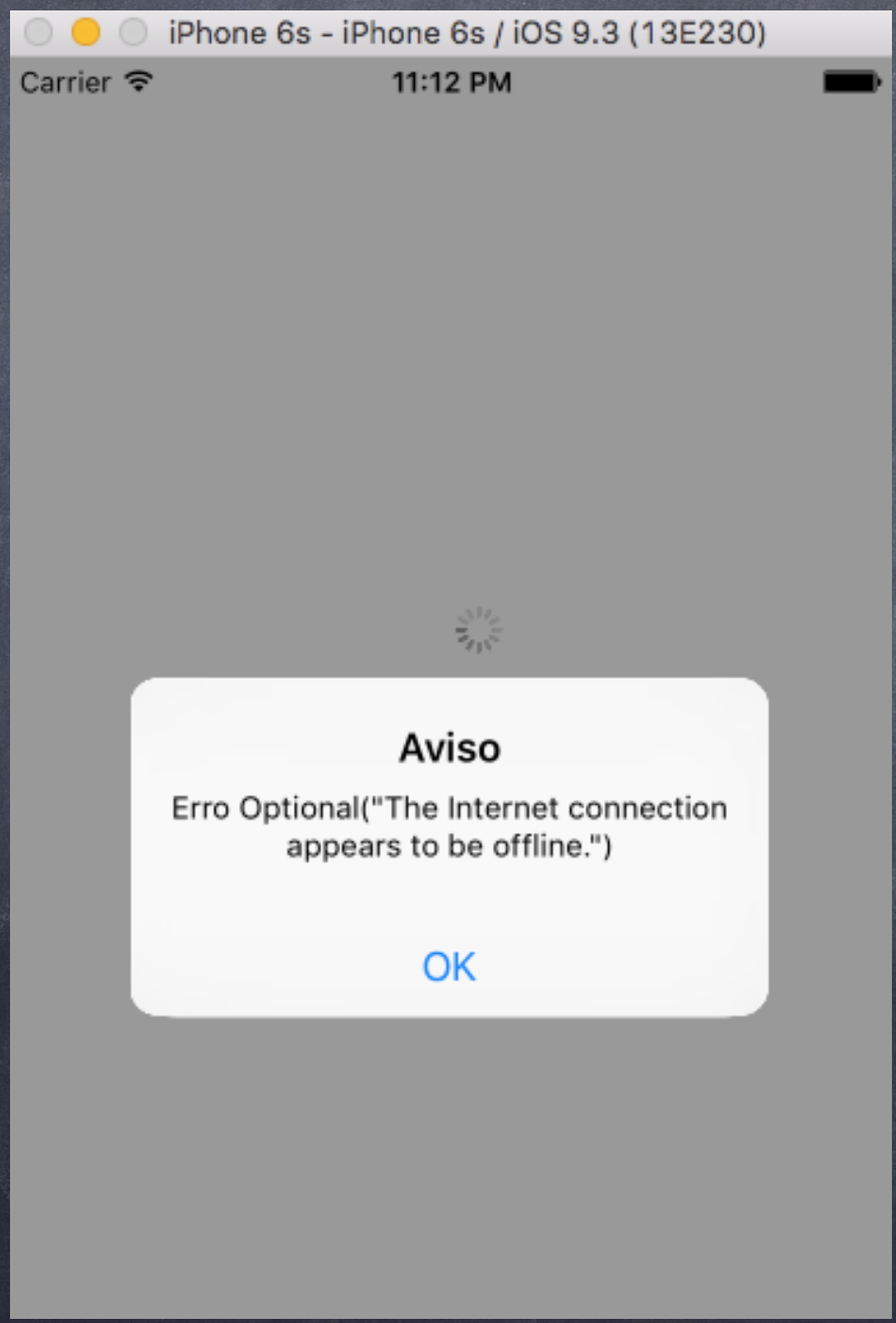
```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
</dict>
```

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.

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

Esconder o Activity

- Existe duas formas de esconder o Activity

```
57
58      func webViewDidFinishLoad(webView: UIWebView) {
59          meuActivity.stopAnimating()
60          meuActivity.hidden = true
61      }
```

1

```
19      override func viewDidLoad() {
20          super.viewDidLoad()
21
22          meuActivity.startAnimating()
23
24          meuWebView.delegate = self
25          //podemos fazer a linha acima conectando o delegate
26          //pelo connection inspector de forma visual, se ligarmos
27          //dessa forma podemos comentar a linha d código acima.
28          //Faremos isso ao final do slide para msotrar como fazê-lo
29          //de forma visual
30
31          let url = NSURL(string: URL_PAGINA )
32          let request = NSURLRequest(URL: url!)
33          meuWebView.loadRequest(request)
34          meuActivity.hidesWhenStopped=true //outra forma com este comando
35          //não precisa da chamada do hidden no webViewDidFinishLoad
36      }
```

2

WebView delegate

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

WebView

- Para implementar o protocolo UIWebViewDelegate visualmente selecione o WebView, em connections inspector (1), ligue o delegate ao ViewController arrastando do radio button(2) até o ViewController(3). 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.