

WebView

Utilizando StoryBoard

X-Code com ObjC

Prof. Agesandro Scarpioni

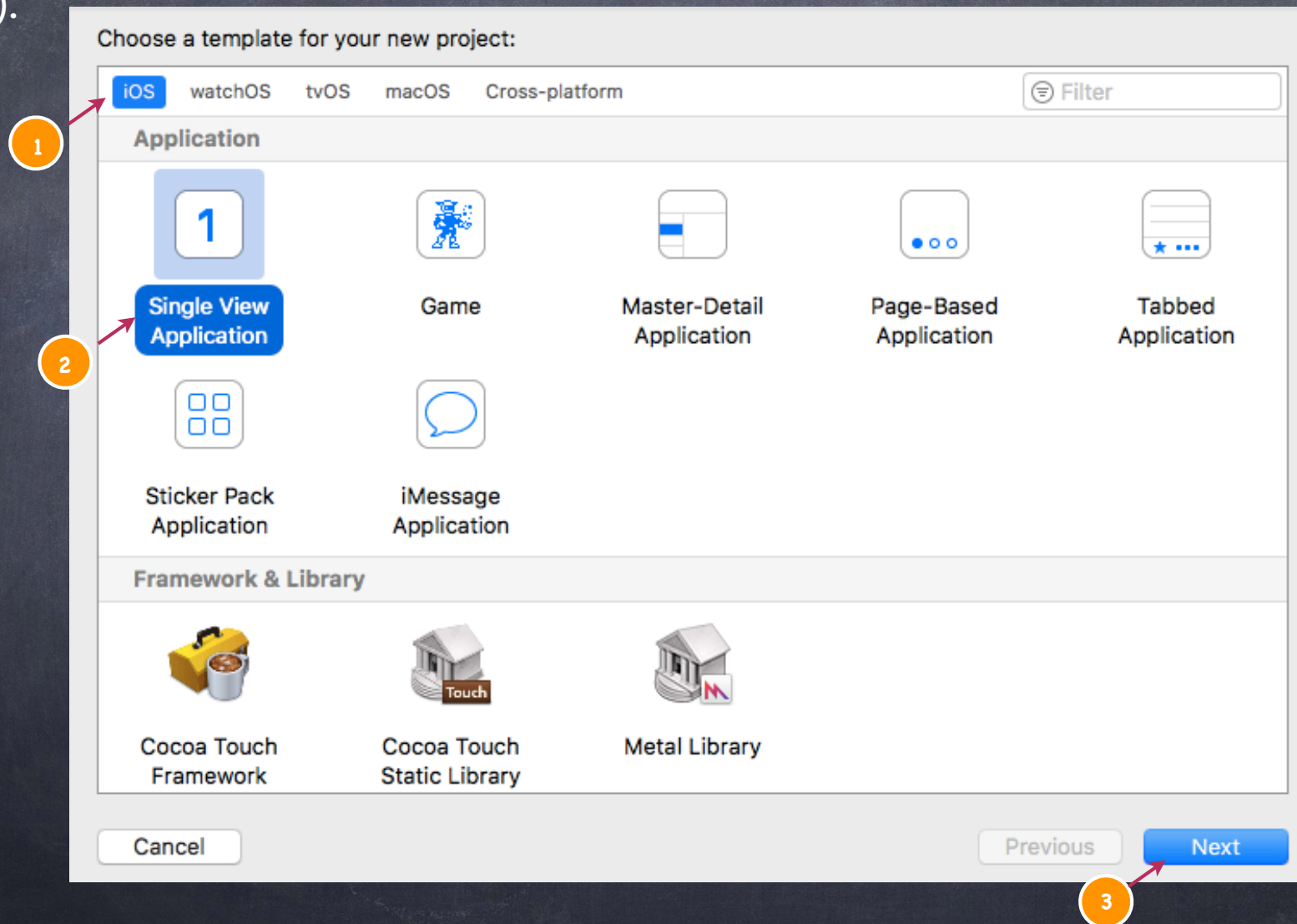
agesandro@fiap.com.br

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_ObjC" (4), selecione a linguagem Objective-C (5) e o device iPhone(6).

Choose options for your new project:

4 → Product Name:

Team:

Organization Name:

Organization Identifier:

Bundle Identifier:

5 → Language:

6 → Devices:

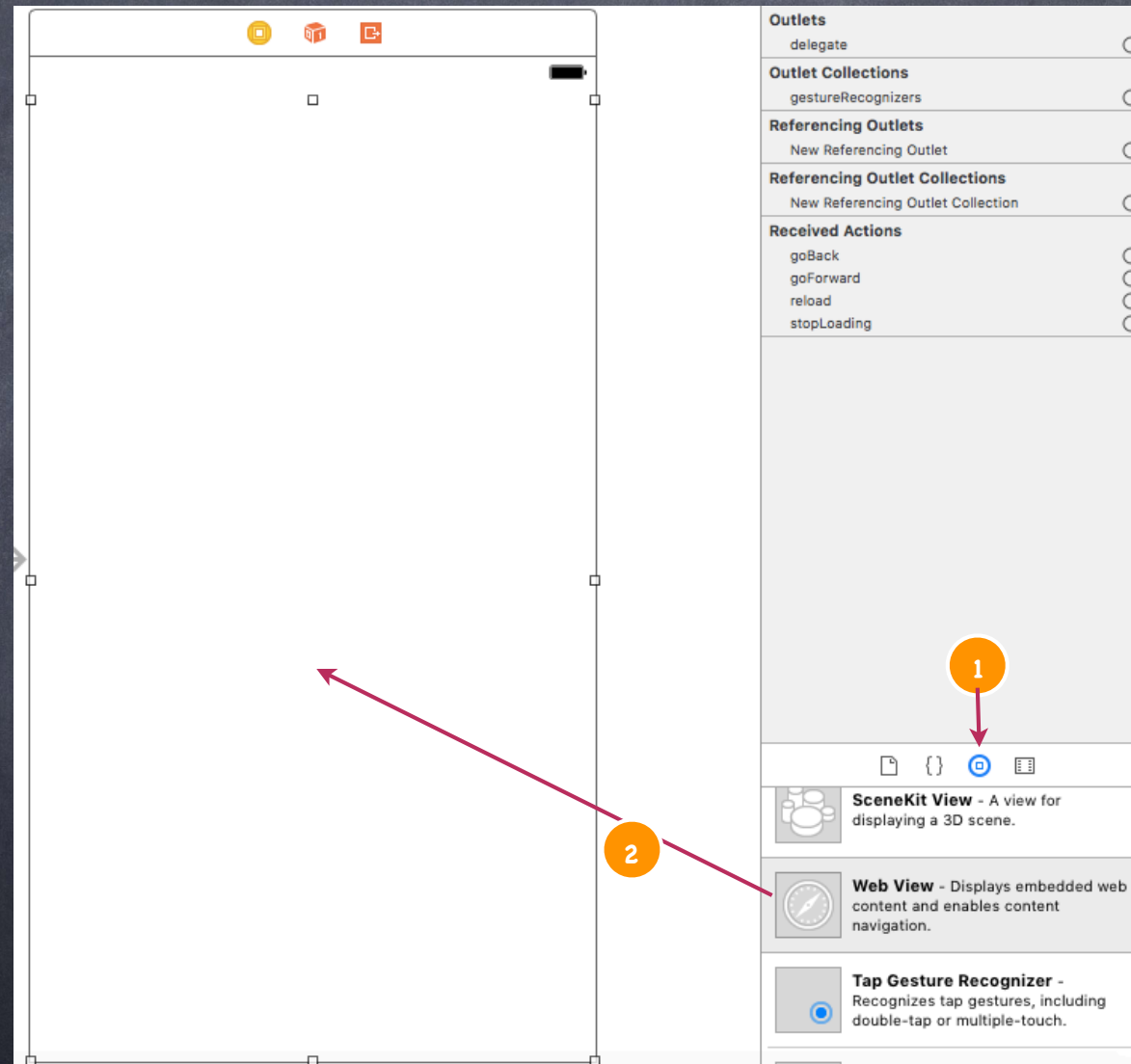
☐ Use Core Data

☐ Include Unit Tests

☐ Include UI Tests

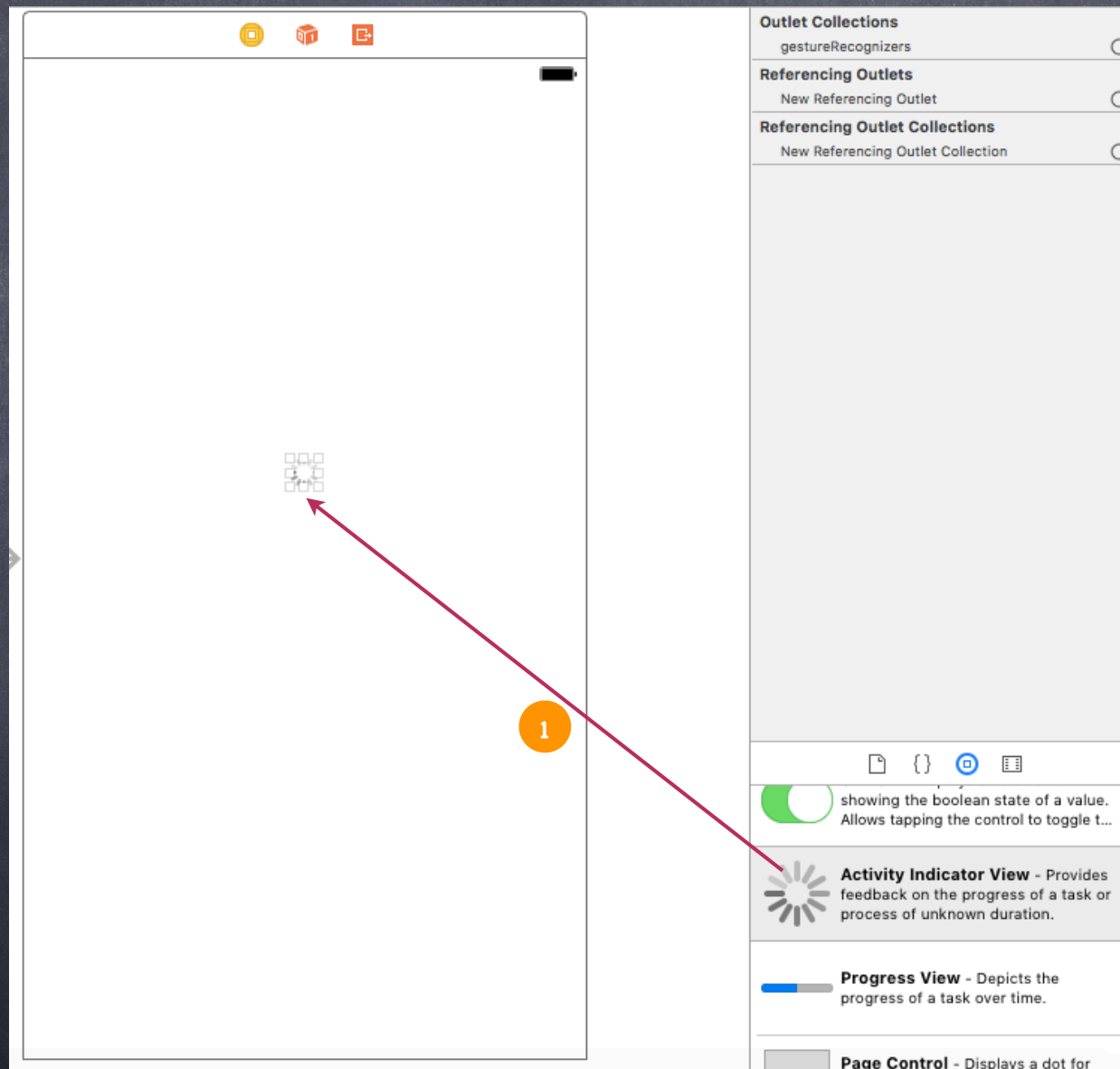
WebView

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



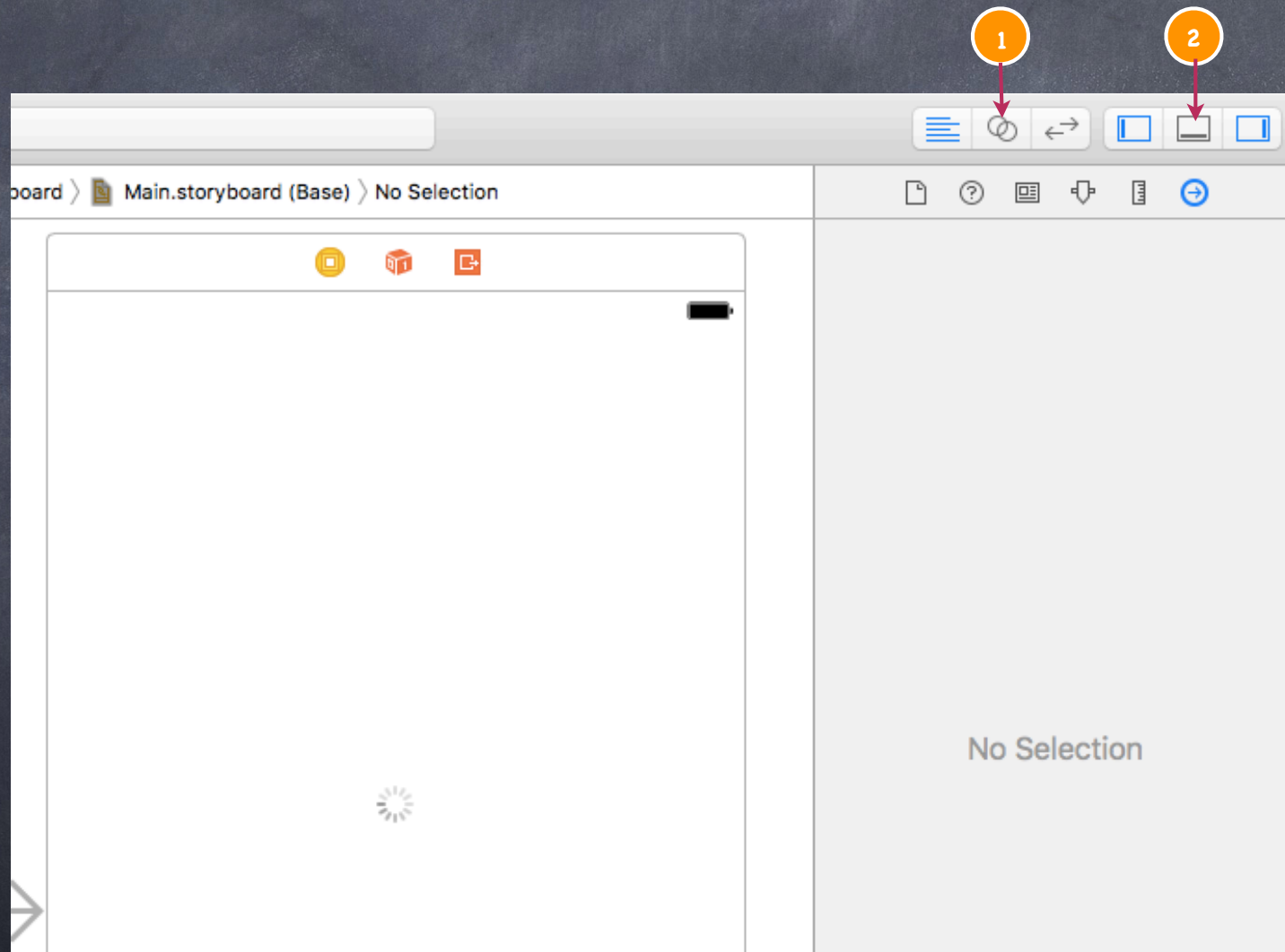
WebView

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



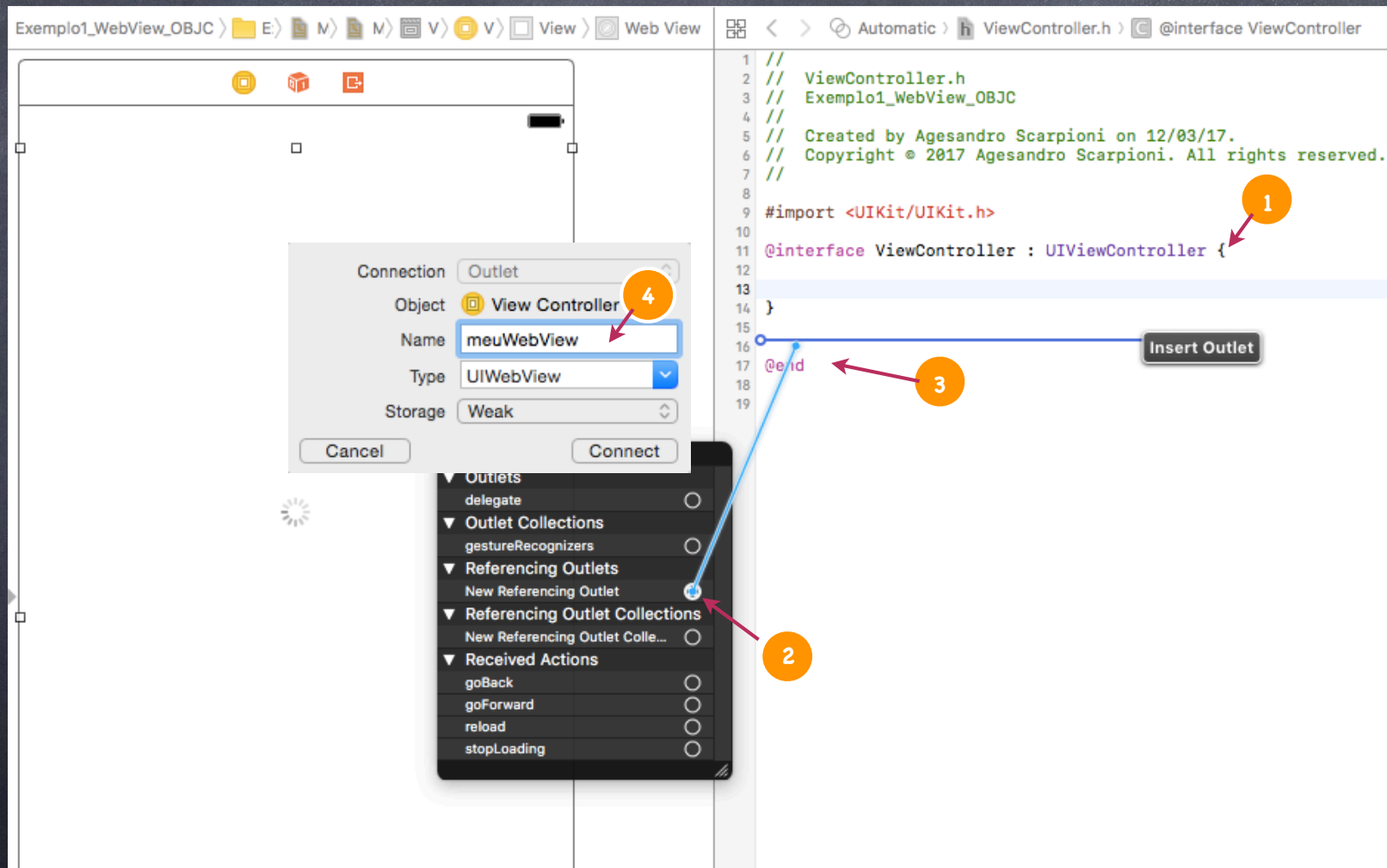
WebView

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



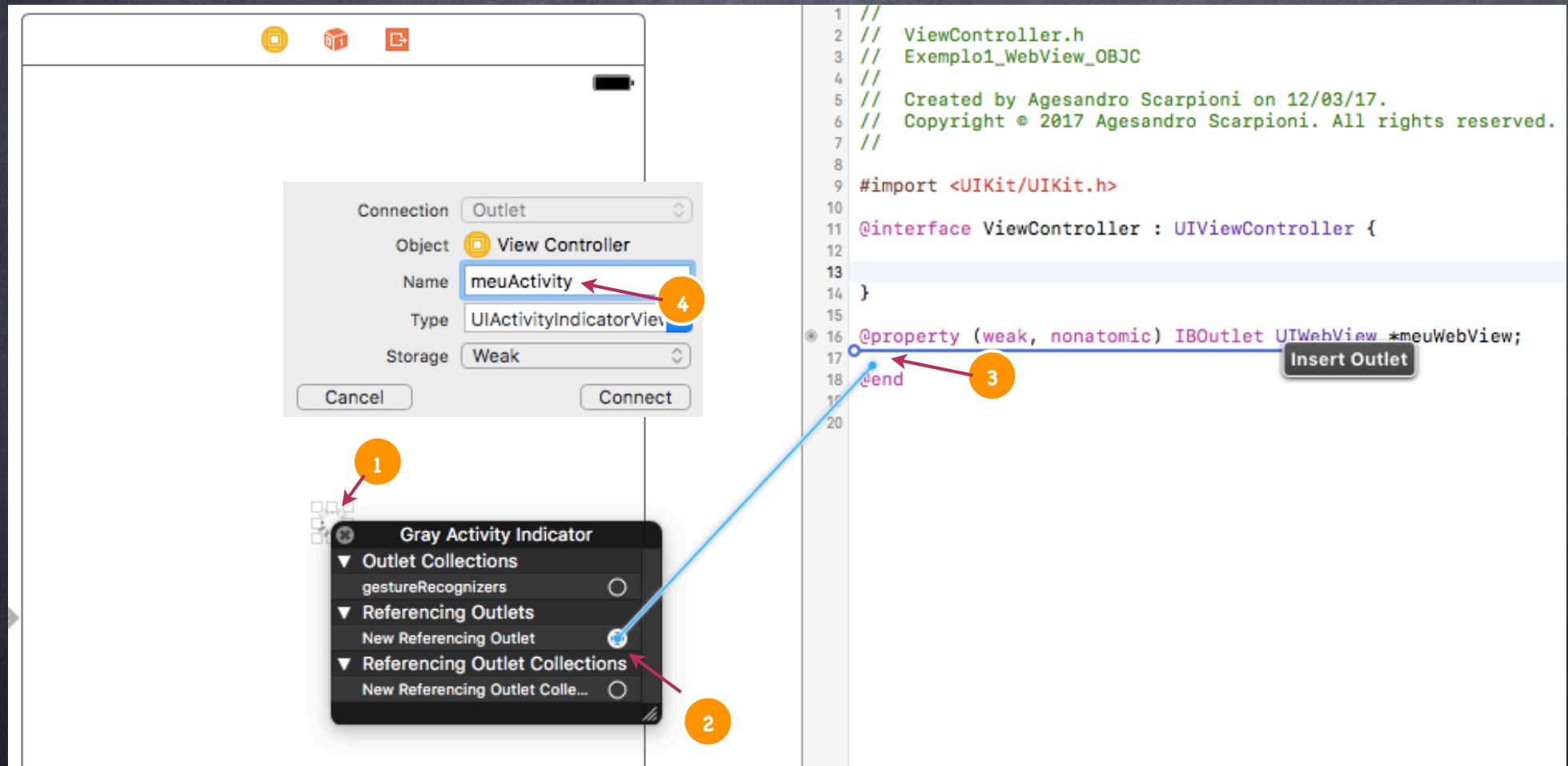
WebView

- Na classe ViewController.h coloque as chaves(1), clique com o botão direito sobre a UIWebView e selecione New Referencing Outlet(2), arraste até a área abaixo das chaves (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).



The image shows the Xcode interface for creating an IBOutlet. On the left, the 'Connect Outlets' menu is open, showing a blue dot next to 'New Referencing Outlet' (labeled 2). A blue line connects this dot to the 'meuActivity' field in the 'Connect Outlets' dialog box (labeled 4). The dialog box shows 'Connection' as 'Outlet', 'Object' as 'View Controller', 'Name' as 'meuActivity', 'Type' as 'UIActivityIndicatorView', and 'Storage' as 'Weak'. On the right, the code for 'ViewController.h' is shown. The code includes the following lines:

```
1 //  
2 // ViewController.h  
3 // Exemplo1_WebView_OBJC  
4 //  
5 // Created by Agesandro Scarpioni on 12/03/17.  
6 // Copyright © 2017 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 #import <UIKit/UIKit.h>  
10  
11 @interface ViewController : UIViewController {  
12  
13 }  
14  
15  
16 @property (weak, nonatomic) IBOutlet UIWebView *meuWebView;  
17 @end
```

The code is annotated with a blue line and a label '3' pointing to the 'IBOutlet' keyword in the property declaration. A button labeled 'Insert Outlet' is also visible next to the property declaration.

WebView

- No ViewController.h 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.h  
3 // Exemplo1_WebView_OBJC  
4 //  
5 // Created by Agesandro Scarpioni on 12/03/17.  
6 // Copyright © 2017 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 #import <UIKit/UIKit.h>  
10  
11 @interface ViewController : UIViewController <UIWebViewDelegate> {  
12  
13  
14 }  
15  
16 @property (weak, nonatomic) IBOutlet UIWebView *meuWebView;  
17 @property (weak, nonatomic) IBOutlet UIActivityIndicatorView *meuActivityIndicator;  
18  
19 @end  
20
```


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: < protocol1, protocol2 > quando a classe implementar mais que um protocolo eles devem ser separados por vírgula.

```
12 @interface Atleta : NSObject <AlimentoLiquidoeSolido>
```

- Abaixo um exemplo de um protocolo que foi criado como AlimentoLiquido e que o método beberIsotonico aguarda uma implementação na classe Atleta.

```
9 #import <Foundation/Foundation.h>
10
11 @protocol AlimentoLiquido <NSObject>
12
13 -(void) beberIsotonico;
14
15 @end
16
```


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 < protocol1 >
 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.

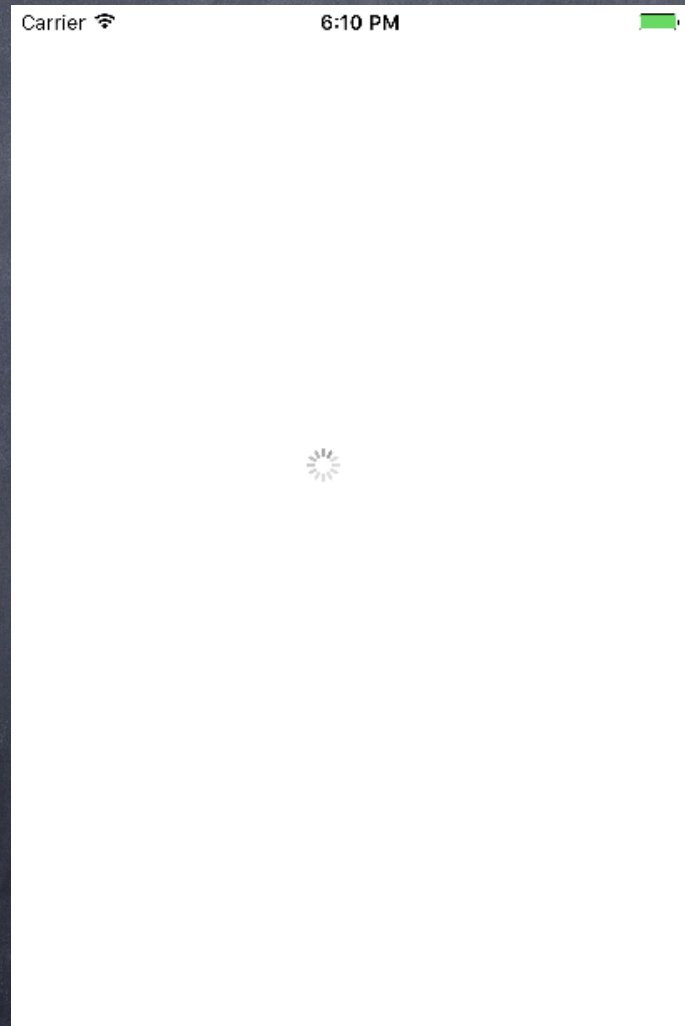
WebView

- No ViewController.m faça o @synthesize (1) das duas propriedades no ViewController.h, inicialize o Activity com startAnimating (2) e declare o protocolo UIWebViewDelegate para a ViewController (3), também é possível fazer essa ligação de forma visual pelo Connections Inspector, veja a página 29.

```
1 //
2 // ViewController.m
3 // Exemplo1_WebView_OBJC
4 //
5 // Created by Agesandro Scarpioni on 12/03/17.
6 // Copyright © 2017 Agesandro Scarpioni. All rights reserved.
7 //
8
9 #import "ViewController.h"
10
11 @interface ViewController ()
12
13 @end
14
15 @implementation ViewController
16
17 @synthesize meuWebView, meuActivity;
18
19 - (void)viewDidLoad {
20     [super viewDidLoad];
21     [self.meuActivity startAnimating];
22
23     [self.meuWebView setDelegate:self]; //podemos fazer isso ligando
24     //o delegate com o WebView pelo Connection Inspector, se ligarmos
25     //dessa forma podemos comentar essa linha de comando acima, faremos
26     //isso ao final do slide para mostrar como ligar o delegate ao
27     //WebView via Connection Inspector
28 }
29
30 - (void)didReceiveMemoryWarning {
31     [super didReceiveMemoryWarning];
32     // Dispose of any resources that can be recreated.
33 }
34
35
36
37 @end
38
```


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 utilizando a diretiva #define para armazenar uma URL.

```
1 //
2 // ViewController.m
3 // Exemplo1_WebView_OBJC
4 //
5 // Created by Agesandro Scarpioni on 12/03/17.
6 // Copyright © 2017 Agesandro Scarpioni. All rig
7 //
8
9 #import "ViewController.h"
10
11 @interface ViewController ()
12
13 @end
14
15 #define URL_PAGINA @"http://www.vive.com/us"
16
17 @implementation ViewController
18
19 @synthesize meuWebView, meuActivity;
20
21 - (void)viewDidLoad {
```



OBS: Existe um Objeto chamado searchBar que é utilizado com um WebView para que seja criado um navegador

WebView

- Ainda no viewDidLoad do viewController.m, crie um objeto URL e um objeto URLRequest, carregue este URLRequest na WebView como é demonstrado abaixo(1).

```
20
21 - (void)viewDidLoad {
22     [super viewDidLoad];
23     [self.meuActivity startAnimating];
24
25     [self.meuWebView setDelegate:self]; //podemos fazer isso ligando
26     //o delegate com o WebView pelo Connection Inspector, se ligarmos
27     //dessa forma podemos comentar essa linha de comando acima, faremos
28     //isso ao final do slide para mostrar como ligar o delegate ao
29     //WebView via Connection Inspector
30
31     //Criamos um objeto url
32     NSURL *url = [NSURL URLWithString:URL_PAGINA];
33     NSURLRequest *request = [NSURLRequest requestWithURL:url];
34     //chamando o método loadRequest
35     [self.meuWebView loadRequest:request];
36
37 }
```

1

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.h` declaramos o protocolo `< UIWebViewDelegate >`

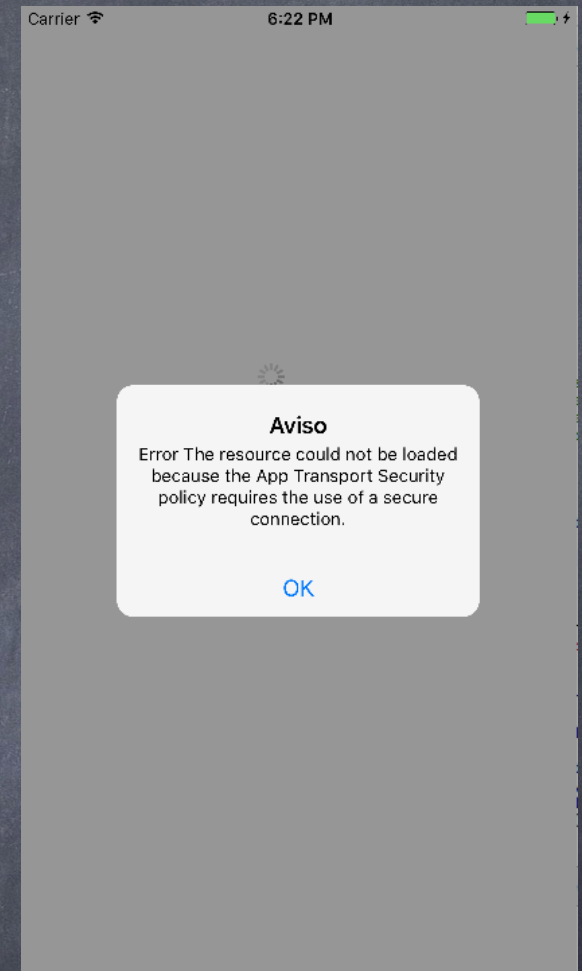
```
29 -(void) webView:(UIWebView *)webView didFailLoadWithError:(NSError *)error{
30     NSString *msg = [NSString stringWithFormat:@"Erro %@", [error localizedDescription]];
31
32     UIAlertController *alerta = [UIAlertController
33                                 alertControllerWithTitle:@"Aviso"
34                                 message:msg
35                                 preferredStyle:UIAlertControllerStyleAlert];
36
37     UIAlertAction *ok = [UIAlertAction
38                         actionWithTitle:@"OK"
39                         style:UIAlertActionStyleDefault
40                         handler:^(UIAlertAction * _Nonnull action) {
41                             [alerta dismissViewControllerAnimated:YES completion:nil];
42                         }];
43
44     [alerta addAction:ok];
45     [self presentViewController:alerta animated:YES completion:nil];
46
47 }
```

- 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

- Você pode adicionar exceções 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>
```

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 em [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

FIAP

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

Info.plist (forma 1) ^{FIA/P}

- 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

The screenshot illustrates the process of adding a new key to the Info.plist file in Xcode. The interface is divided into three main sections: the Project Navigator on the left, the Key-Value List in the center, and the Key-Value Editor on the right.

Project Navigator (Left): Shows the project structure for 'Exemplo1_WebView_Swift'. The 'Info.plist' file is highlighted with a red arrow and a circled '1'.

Key-Value List (Center): A table listing various keys and their types. The 'Application Category' key is selected, and a red arrow with a circled '2' points to the '+' button next to it.

Key-Value Editor (Right): Shows the details for the selected key. A red arrow with a circled '3' points to the 'App Transport Security Settings' option in the dropdown menu. Another red arrow with a circled '4' points to the triangle icon next to the 'App Transport Security Settings' key in the list. A third red arrow with a circled '5' points to the '+' button next to the 'App Transport Security Settings' key in the list. A fourth red arrow with a circled '6' points to the 'Allow Arbitrary Loads' key in the list. A fifth red arrow with a circled '7' points to the 'YES' value for the 'Allow Arbitrary Loads' key.

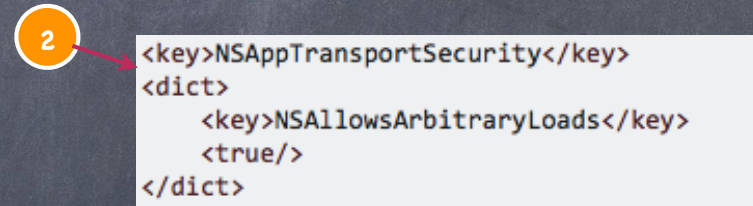
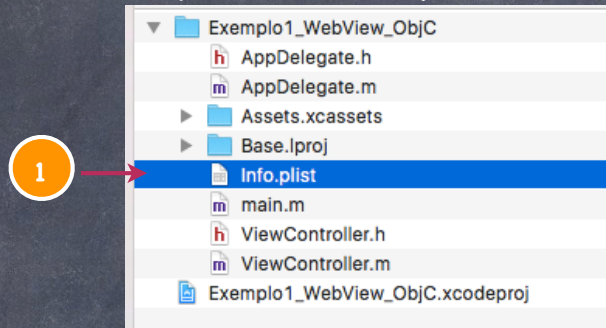
Key	Type
Information Property List	Dictionary
Localization native development re...	String
Executable file	String
Bundle identifier	String
InfoDictionary version	String
Bundle name	String
Bundle OS Type code	String
Bundle versions string, short	String
Bundle creator OS Type code	String
Bundle version	String
Application requires iPhone enviro...	Boolean
Launch screen interface file base...	String
Main storyboard file base name	String
Required device capabilities	Array
Supported interface orientations	Array
Application Category	String

Key	Type	Value
App Transport Security Settings	Dictionary	(0 items)
App Uses Non-Exempt Encrypti...	String	
Application can be killed immed...	String	
Application Category	String	
Application does not run in bac...	String	
Application fonts resource path	String	
Application has localized displa...	String	
Application is agent (UIElement)	String	
Application is background only	String	
Application is visible in Classic	String	

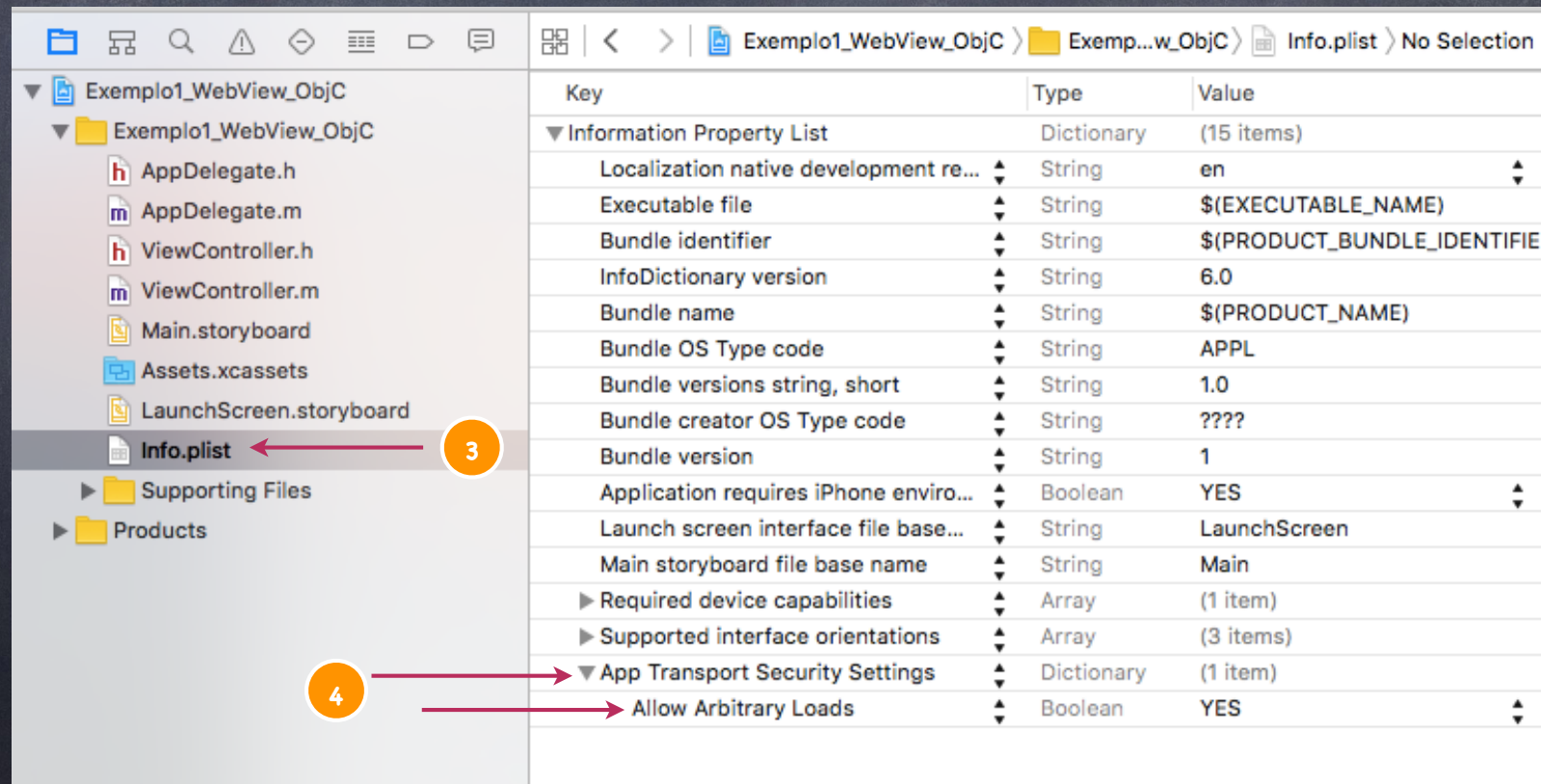
Key	Type	Value
App Transport Security Settings	Dictionary	(1 item)
Allow Arbitrary Loads	Boolean	YES

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).

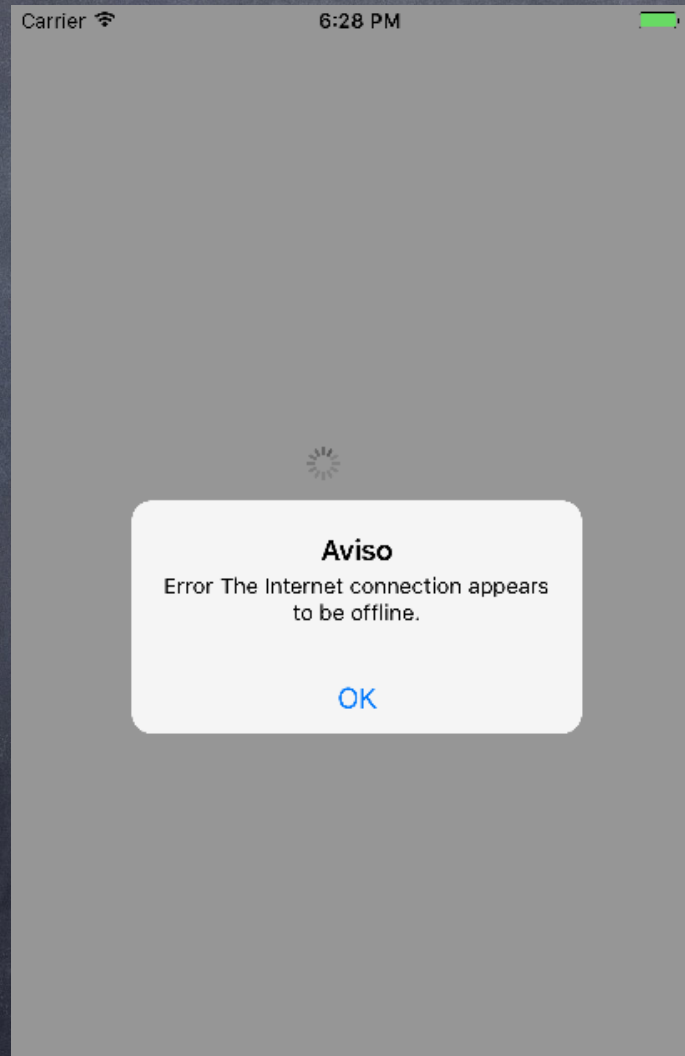


Dica: A forma 2 é ideal quando você possui o sinalizador no formato de tag < >



WebView delegate FIAP

- 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 FIA.P

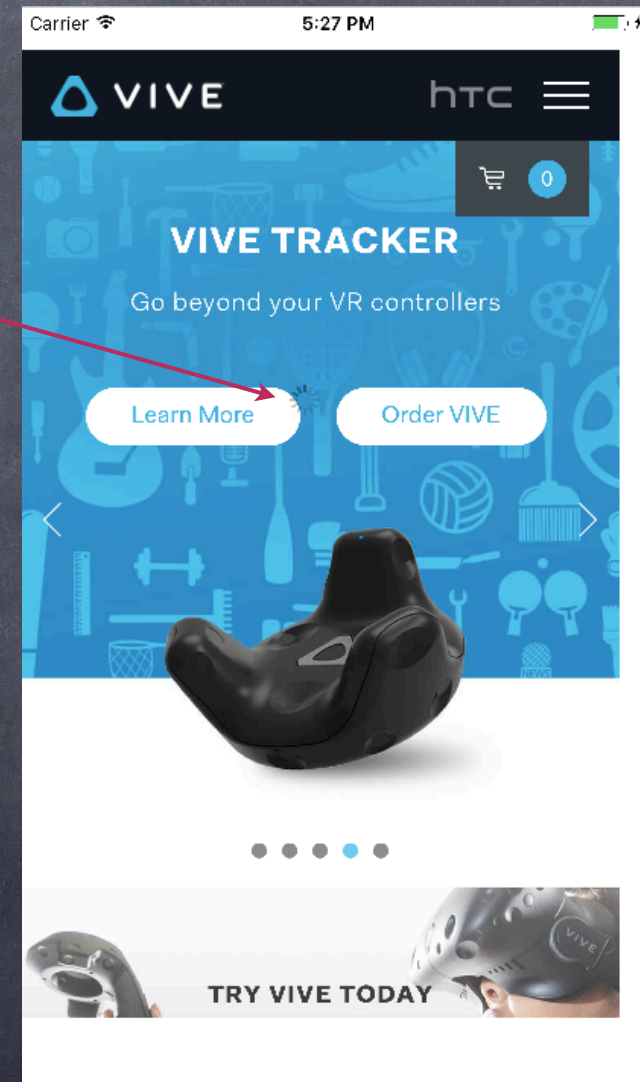
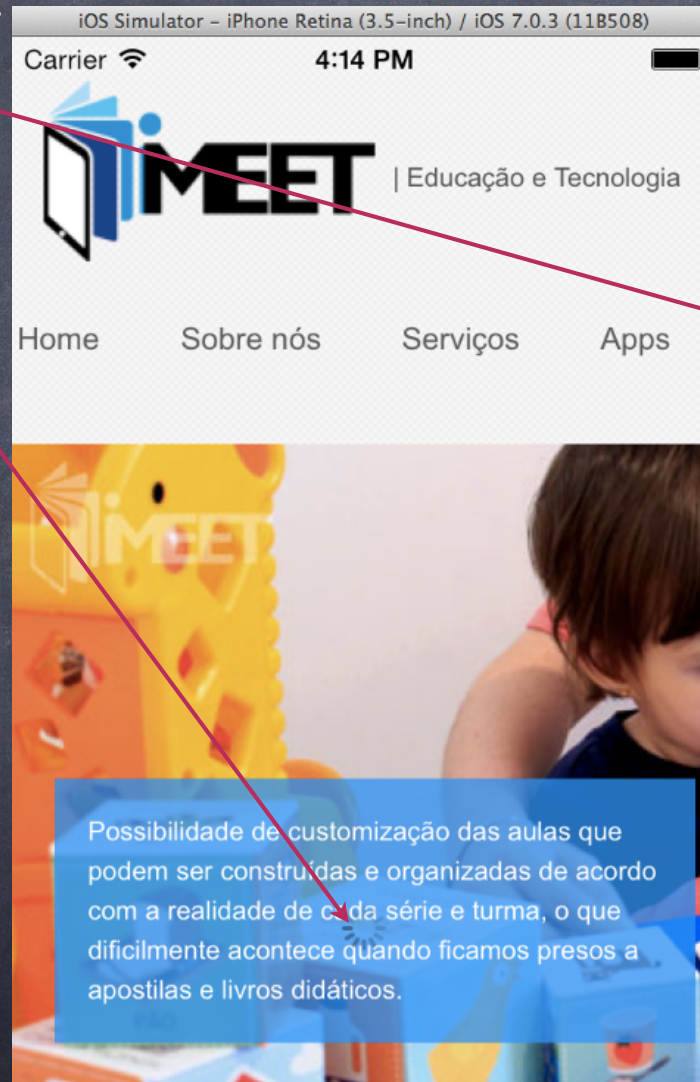
- O método `webViewDidFinishLoad` do protocolo `UIWebViewDelegate` é chamado no final da requisição, quando uma página é carregada com sucesso.

```
46  
47 -(void) webViewDidFinishLoad:(UIWebView *)webView {  
48     // este método é chamado no final da requisição  
49     //quando a página foi carregada com sucesso.  
50     [meuActivity stopAnimating];  
51 }
```


WebView delegate

FIAP

- 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

```
40
41 -(void) webViewDidFinishLoad:(UIWebView *)webView {
42     //este método é chamado no final da requisição quando a página foi carregada com sucesso
43     [meuActivity stopAnimating];
44     1 → [meuActivity setHidden:YES];
45 }
46
47
```

```
20
21 - (void)viewDidLoad {
22     [super viewDidLoad];
23     //iniciando o activity indicator
24     [self.meuActivity startAnimating];
25
26     [self.meuWebView setDelegate:self]; //podemos fazer isso ligando
27     // o delegate com o WebView pelo connection inspector, se ligarmos
28     // dessa forma podemos comentar essa linha de comando, faremos
29     // isso ao final do slide para mostrar como ligar o delegate ao
30     // Webview via Connection Inspector.
31
32     //Criamos um objeto url
33     NSURL *url = [NSURL URLWithString:URL_PAGINA];
34     NSURLRequest *request = [NSURLRequest requestWithURL:url];
35     //chamando o método loadRequest
36     [self.meuWebView loadRequest:request];
37     2 → [self.meuActivity setHidesWhenStopped:YES]; //outra forma, este modelo não precisa
38     //da chamada no didFinishload
39
40     // Do any additional setup after loading the view, typically from a nib.
41 }
42
```


WebView delegate FIAP

- 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 `"[self.meuWebView setDelegate:self];"` no `viewDidLoad` pode ser comentada, ou seja, existem duas formas de associar o protocolo, pelo código ou de forma visual pelo `connections inspector`.

