

Rest e Json no iOS

X-Code com Swift
Prof. Agesandro Scarpioni

App's com label's, image's e button's

- Criar dois aplicativos para que seja feito um request, depois um parser do Json em um dicionário para exibir os dados.

O que é Rest

- REST, abreviação de "REpresentational State Transfer" (Transferência de Estado Representativo) é uma técnica de engenharia de Software para sistemas hipermídia distribuídos como World Wide Web. REST é um estilo de se projetar aplicativos da Web fracamente acoplados que contam com recursos nomeados em forma de Localizador Uniforme de Recursos (URL), Identificador Uniforme de Recursos (URI) e Nome de Recurso Uniforme (URN), e não com mensagens. Engenhosamente, o REST transporta a infra estrutura já validada e bem sucedida da Web, HTTP, ou seja, o REST alavanca aspectos do protocolo HTTP como pedidos GET e POST. Esses pedidos são perfeitamente mapeados para necessidades de aplicativo de negócios padrão, como create read, update, and delete (CRUD).
- Na arquitetura REST os clientes (páginas web, dispositivos iOS, Android) efetuam requisições (request) através do protocolo HTTP e recebem respostas (response) dos servidores. Dessa forma é feita a transmissão de dados.
- Utilizar um Webservice é uma das maneiras mais comuns de se integrar aplicações diferentes. Existem diferentes tipos de arquiteturas para web services, e o RESTful é mais simples em comparação aos outros web services, que geralmente utilizam o SOAP.

Dica: Saiba mais sobre Rest em: <http://www.ibm.com/developerworks/br/library/j-rest/>

O que é Rest

- Dada essa simplicidade, isso faz com que a arquitetura RESTful seja uma escolha popular principalmente para serviços abertos ao público. Por exemplo, o Twitter possui uma API Restful. Além do Twitter, o Flickr também possui uma API que segue os princípios da arquitetura REST. De certa forma é uma tendência que os serviços conhecidos como a “Web 2.0” disponibilizem uma API (geralmente REST), pois é cada vez maior a necessidade que esses serviços sejam integrados com diversos tipos aplicações.

Dica: Saiba mais sobre Rest em: <http://imasters.com.br/desenvolvimento/definicao-restricoes-e-beneficios-modelo-de-arquitetura-rest/>

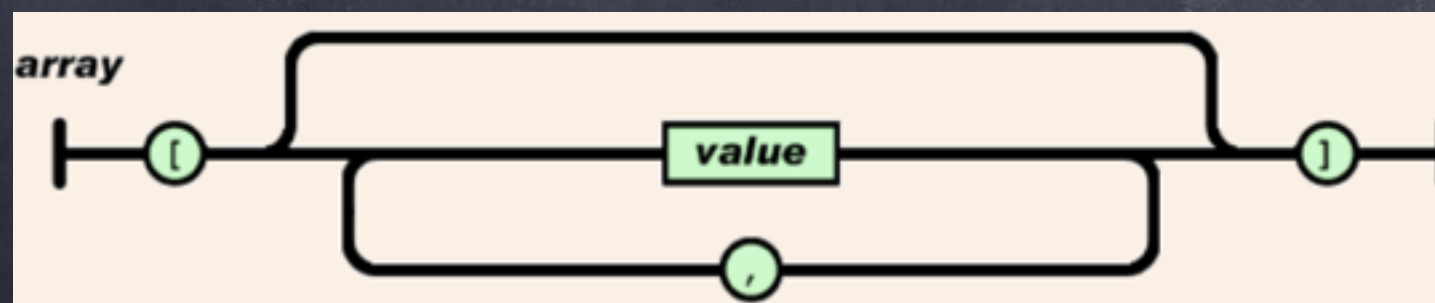
O que é JSON

- **JSON** É um acrônimo para **JavaScript Object Notation**, é um formato leve para intercâmbio de dados computacionais, é um subconjunto da notação de objeto de JavaScript, mas seu uso não requer JavaScript exclusivamente. A simplicidade de JSON tem resultado em seu uso difundido, especialmente como uma alternativa para XML em AJAX. Uma das vantagens reivindicadas de **JSON** sobre XML como um formato para intercâmbio de dados, é o fato de ser muito mais fácil escrever um analisador JSON. Na prática esses arquivos Json são retornados de um Web Service. Veja abaixo um exemplo de um objeto JSON.

```
{ "Alunos" : [
  { "nome": "João", "notas": [ 8, 9, 7 ] },
  { "nome": "Maria", "notas": [ 8, 10, 7 ] },
  { "nome": "Pedro", "notas": [ 10, 10, 9 ] }
]
```

Diagrama de anotação: O código JSON acima é exibido com duas anotações em português. Um oval laranja rotulado "Array" aponta com uma seta vermelha para o caractere "[" no início da lista de alunos. Outro oval laranja rotulado "Array" aponta com uma seta vermelha para o primeiro caractere "[" dentro da lista de notas de João.

Um array Json é uma coleção de valores ordenados. O array começa com [e termina com], os valores são separados por vírgula, veja a representação gráfica abaixo:



Dica: Saiba mais sobre JSON em: <http://json.org/json-pt.html>

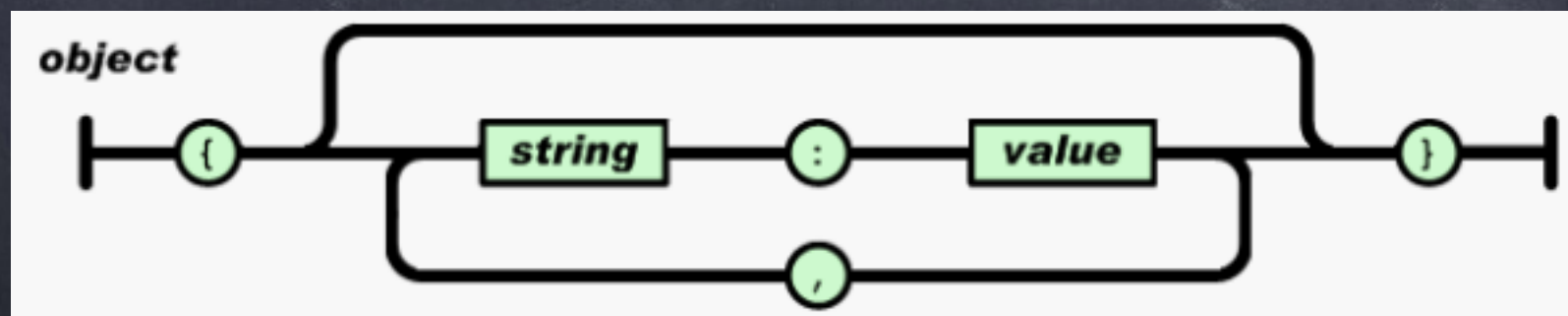
O que é JSON

- JSON é um formato de intercâmbio de dados leve. É fácil para os seres humanos a ler e escrever. É fácil para máquinas para analisar e gerar. É baseado em um subconjunto da linguagem de programação JavaScript. JSON é um formato de texto que é completamente independente do idioma, mas usa convenções que são familiares aos programadores das linguagens da família C, incluindo C, C++, C#, Java, JavaScript, Perl, Python, e muitos outros. Estas propriedades fazem JSON uma linguagem de intercâmbio de dados ideal. Quando você observa um JSON, ele vai ser muito semelhante a um outro exemplo abaixo:

```
{  
  "id": 123,  
  "name": "Jordan G",  
  "age": 17  
}
```

Um objeto JSON é um conjunto não ordenado de pares nome / valor. Um objeto começa com a chave esquerda { e termina com a chave direita }. Cada nome é seguido por dois pontos: e os pares nome / valor são separados por uma vírgula.

Abaixo está uma representação gráfica da forma JSON acima



Dica: Saiba mais sobre JSON em: <http://www.ios-blog.co.uk/tutorials/swift/parse-json-deserialization/#references>

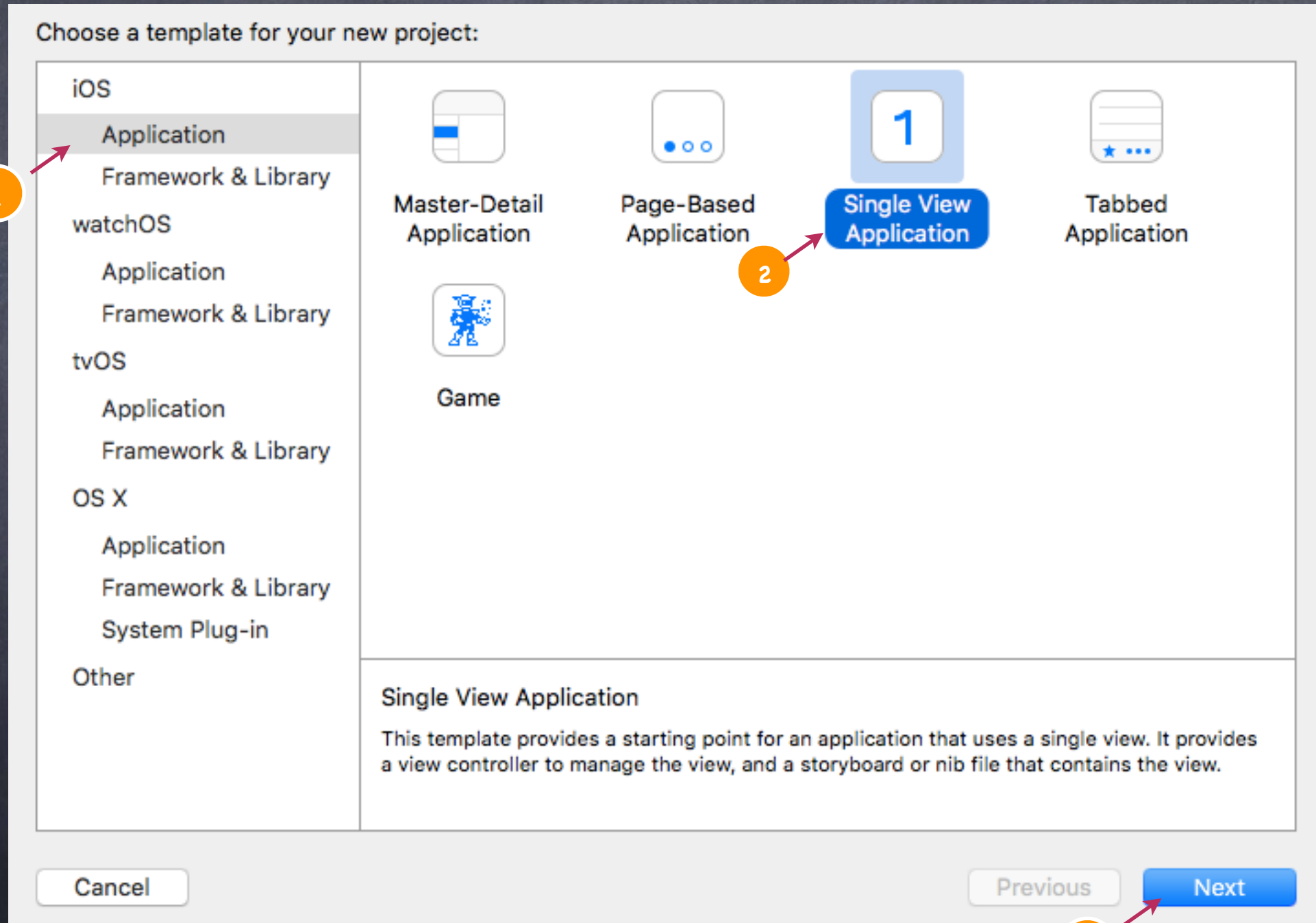
O que é Parser

- Um Parser é um programa de computador ou apenas um componente de um programa que serve para analisar a estrutura gramatical de uma entrada, manipulando os tokens, que são segmentos de texto ou símbolos que podem ser manipulados. Em XML, o parser pode ser um leitor que ajuda na conversão do arquivo para manipulação dos dados contidos no mesmo.
- Para trafegar informações entre o servidor e o aparelho usamos arquivos XML ou JSON, existem dois tipos de Parser de XML: O SAX e o DOM o SAX é o mais conhecido e consome poucos recursos, o SAX é mais trabalhoso que o DOM, O DOM lê o arquivo inteiro em memória criando uma árvore que pode ter suas tags do XML lidas com muita facilidade, O SAX vai navegando pela estrutura do XML linha a linha, e no momento que as tags são encontradas um objeto da aplicação é notificado para ler os dados. O SAX utiliza menos memória que o DOM.
- Pela simplicidade vamos abordar apenas o PARSER do JSON.

Dica: Mais exemplos de JSON em: <http://json.org/example.html>

Iniciando o Projeto

- Clique em File -> New Project -> iOS -> Application -> Single View Application.



Os dados do Projeto

- Preencha com os dados abaixo, lembre-se que o Organization Identifier é como se fosse o pacote no Java ou o namespace do VB, em language escolha Swift, em Devices selecione iPhone.

Choose options for your new project:

4 Product Name: Exemplo2_Rest_iOS_Swift

Organization Name: Agesandro Scarpioni

Organization Identifier: com.scarpioni

Bundle Identifier: com.scarpioni.Exemplo2-Rest-iOS-Swift

5 Language: Swift

6 Devices: iPhone

☐ Use Core Data

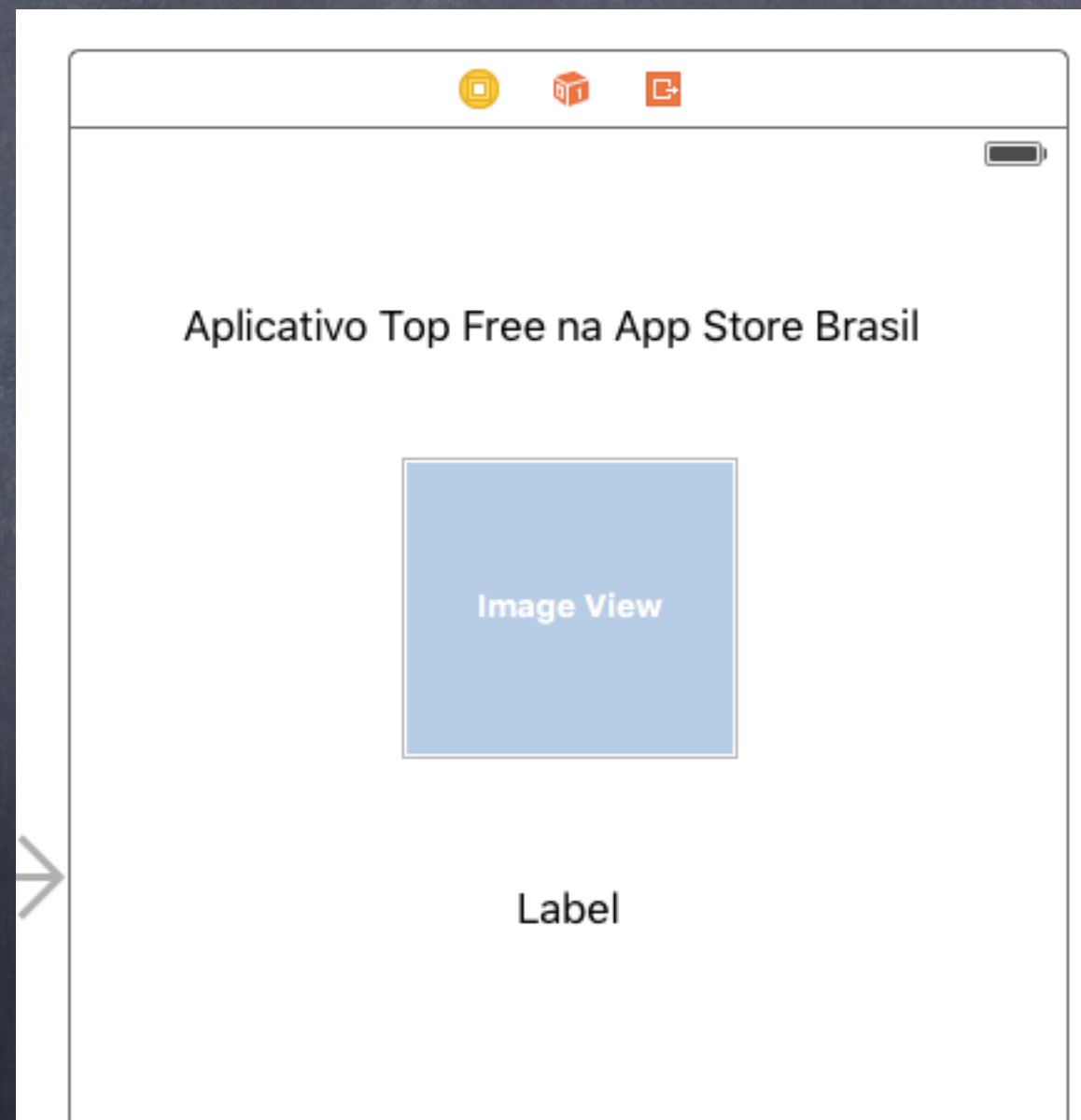
☐ Include Unit Tests

☐ Include UI Tests

Cancel Previous Next

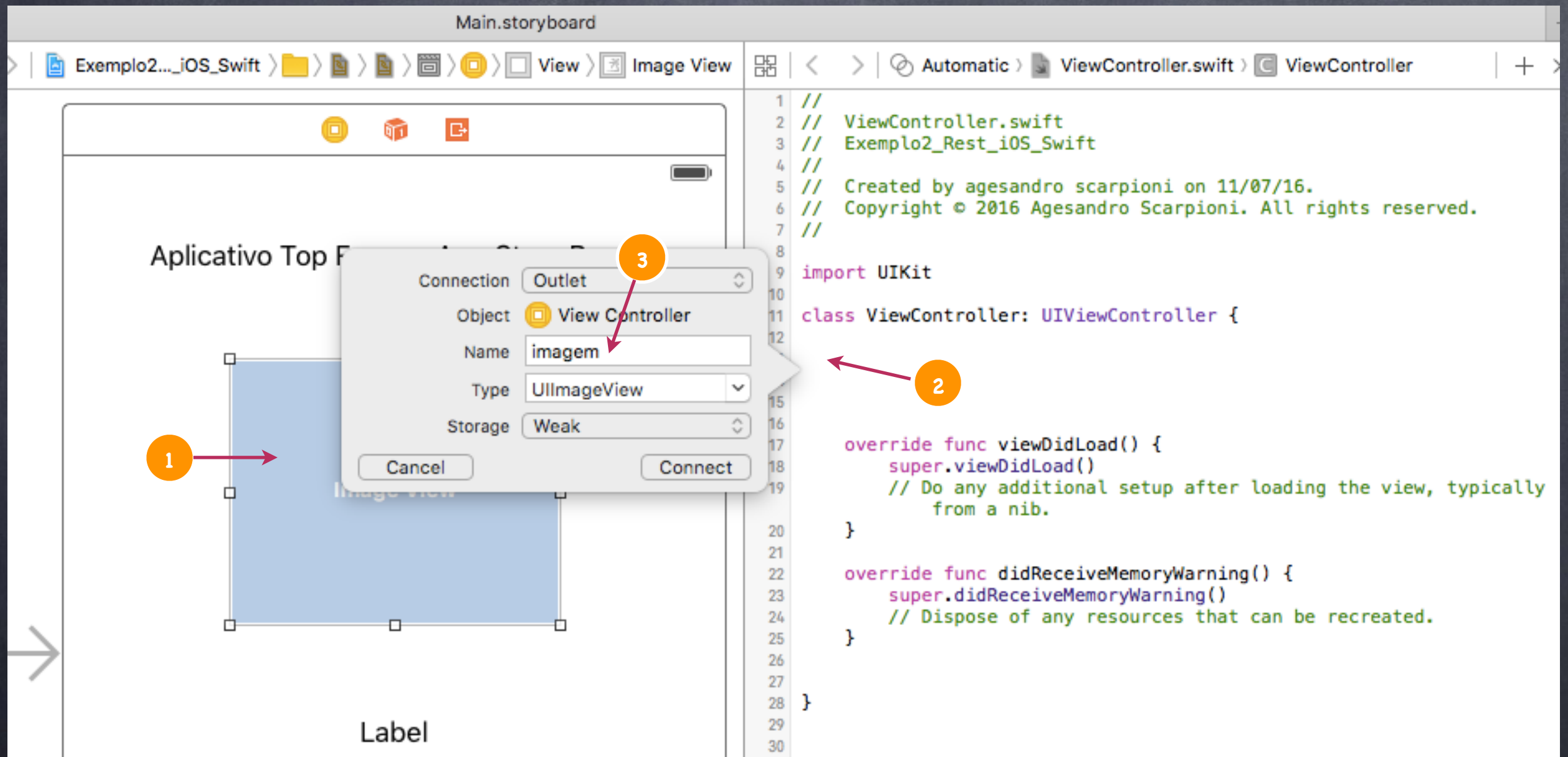
A Interface

- Desenhe a tela abaixo com 2 labels e 1 ImageView.



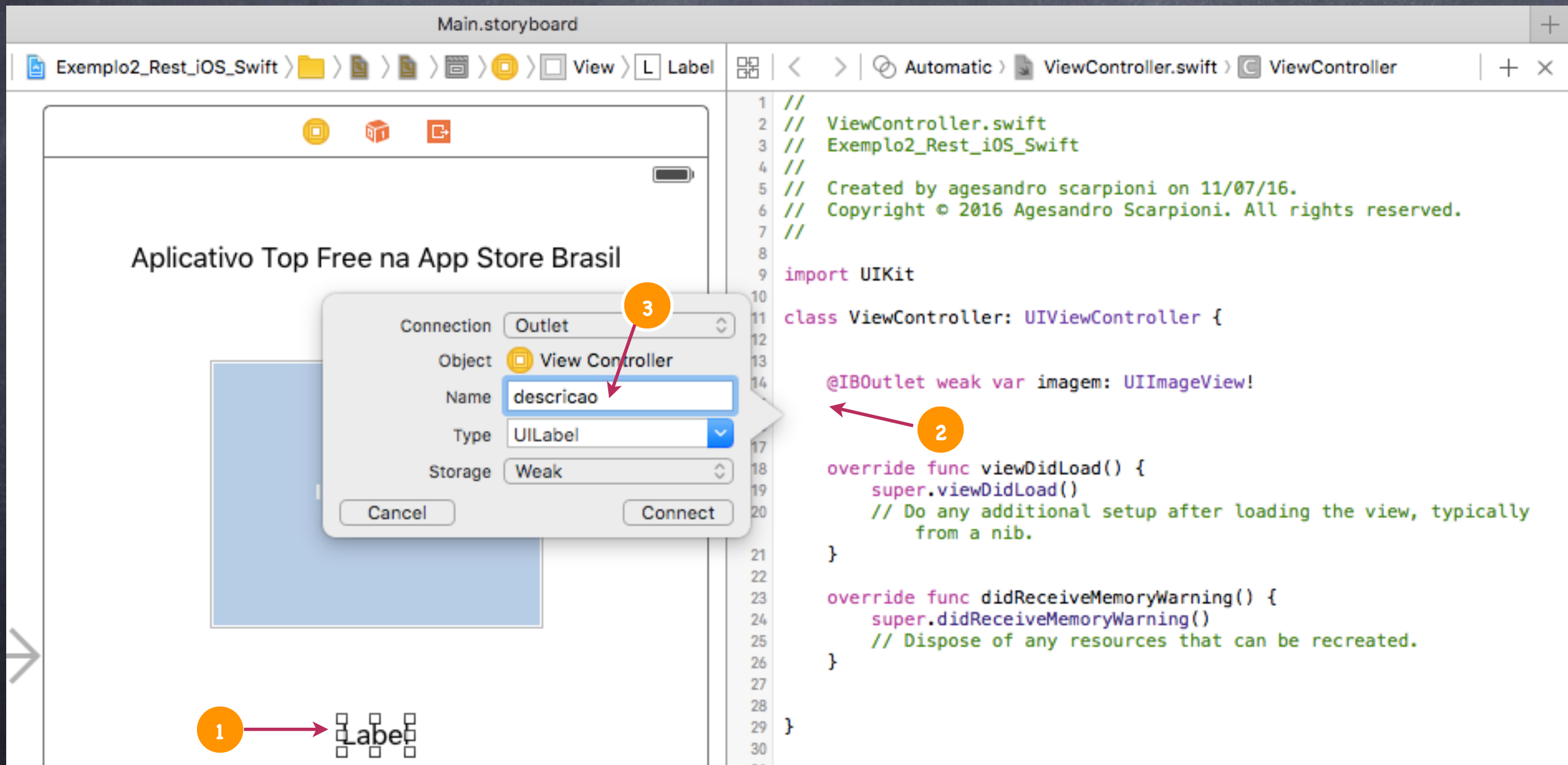
Definindo os IBOutlet's

- ❶ Criar no ViewController.swift os Outlet's do Label e do Image. Selecione o objeto Image (1) e arraste até a área indicada (2), nomeie como imagem (3).



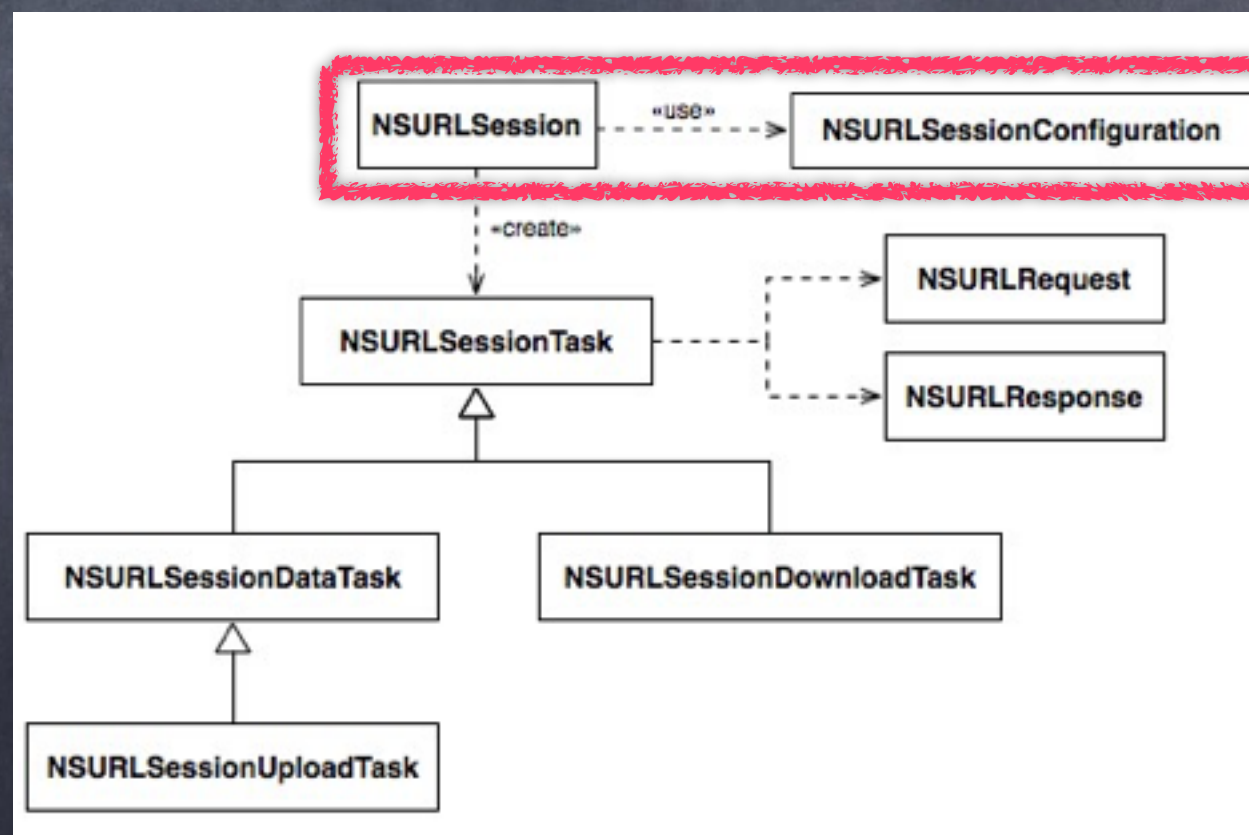
Definindo os IBOutlet's

- Selecione o objeto Label (1) e arraste até a área indicada (2), nomeie como descricao (3).



NSURLSession

- Uma das maneiras de se comunicar com APIs através do http no iOS é utilizar a classe NSURLSession e outras classes relacionadas, a NSURLSession possui características como uploads e downloads em background, pausar e retornar operações em rede, facilitar processo de autenticação etc.
- Uma Session pode ser configurada por meio da classe NSURLSessionConfiguration.



Dica: Mais sobre NSURLSessionConfiguration clique aqui para o site da Apple: [Site da Apple](#)

NSURLSessionConfiguration

- Um objeto da classe `NSURLSessionConfiguration` define o comportamento e as regras a serem utilizadas quando uma sessão efetuar upload e download de dados, como por exemplo:
 1. `allowsCellularAccess`: define se a conexão pode ser feita através da rede de celular
 2. `timeoutIntervalForRequest`: determina o timeout de requests
 3. `HTTPCookieAcceptPolicy`: determina quando cookies devem ser aceitos
 4. `URLCache`: promove cache de respostas dentro de uma sessão

NSURLSessionConfiguration

5. HTTPAdditionalHeaders: cria um dicionário com informações para serem enviadas no header
6. HTTPMaximumConnectionsPerHost: determina o número máximo de conexões simultâneas a um host
7. Método defaultSessionConfiguration(): cria uma configuração padrão que configura entre outras regras um cache persistente em disco e armazena cookies.

Exemplo de uso:

```
var sessionConfig = NSURLSessionConfiguration.defaultSessionConfiguration()  
sessionConfig.allowsCellularAccess = false;  
sessionConfig.HTTPAdditionalHeaders = ["Accept":"application/json"]  
sessionConfig.timeoutIntervalForRequest = 30.0;  
sessionConfig.HTTPMaximumConnectionsPerHost = 1;
```

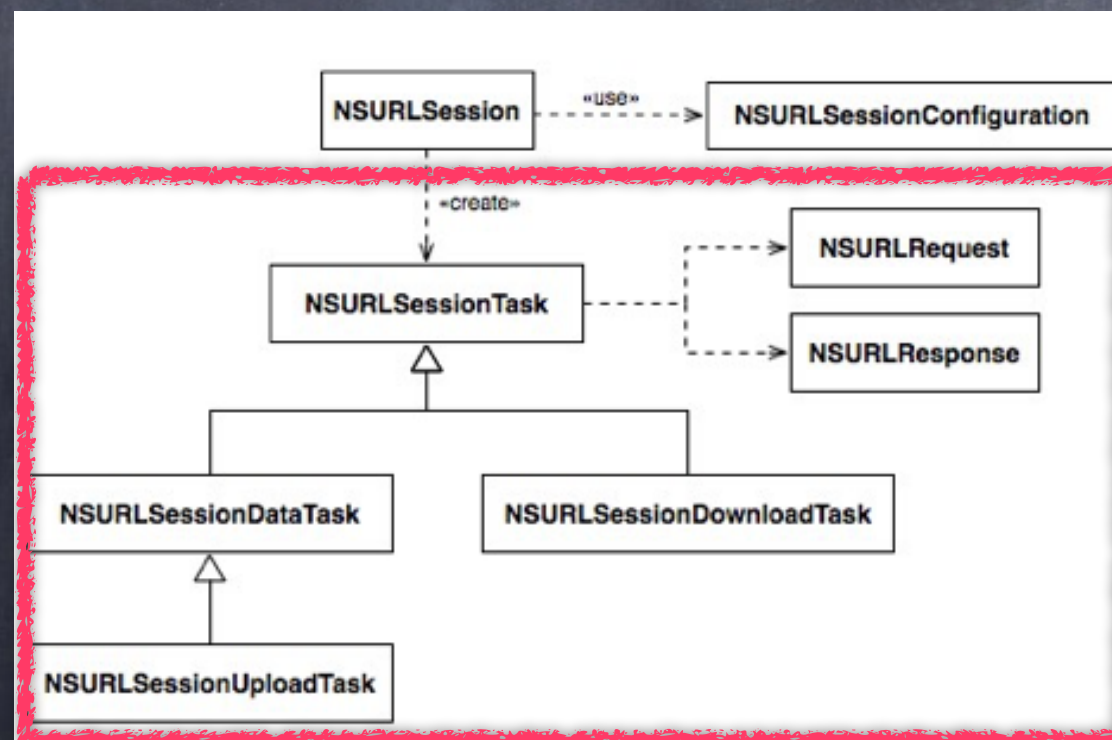

Iniciando uma NSURLSession

- Implemente as linhas 18, 24 27 e 30 para criação e configuração da sessão e definição da URL, a url abaixo é da própria App Store, o termo limit=1 irá trazer o aplicativo grátis top 1 da loja, caso troque o limit para 10 será montado um dicionário com os App's top 10 free, porém, para exibir os dados seria necessário uma lista.

```
1 //
2 // ViewController.swift
3 // Exemplo2_Rest_iOS_Swift
4 //
5 // Created by agesandro scarpioni on 11/07/16.
6 // Copyright © 2016 Agesandro Scarpioni. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController {
12
13
14     @IBOutlet weak var imagem: UIImageView!
15
16     @IBOutlet weak var descricao: UILabel!
17
18     var session: NSURLSession?
19
20     override func viewDidLoad() {
21         super.viewDidLoad()
22
23         //cria um configuracao de sessao default
24         let sessionConfig = NSURLSessionConfiguration.defaultSessionConfiguration()
25
26         //cria uma sessao com a configuracao default
27         session = NSURLSession(configuration: sessionConfig)
28
29         //URL de acesso a API do itunes para retornar o App top Free
30         let url = NSURL (string:"https://itunes.apple.com/br/rss/topfreeapplications/limit=1/json")
31
32     }
33 }
```

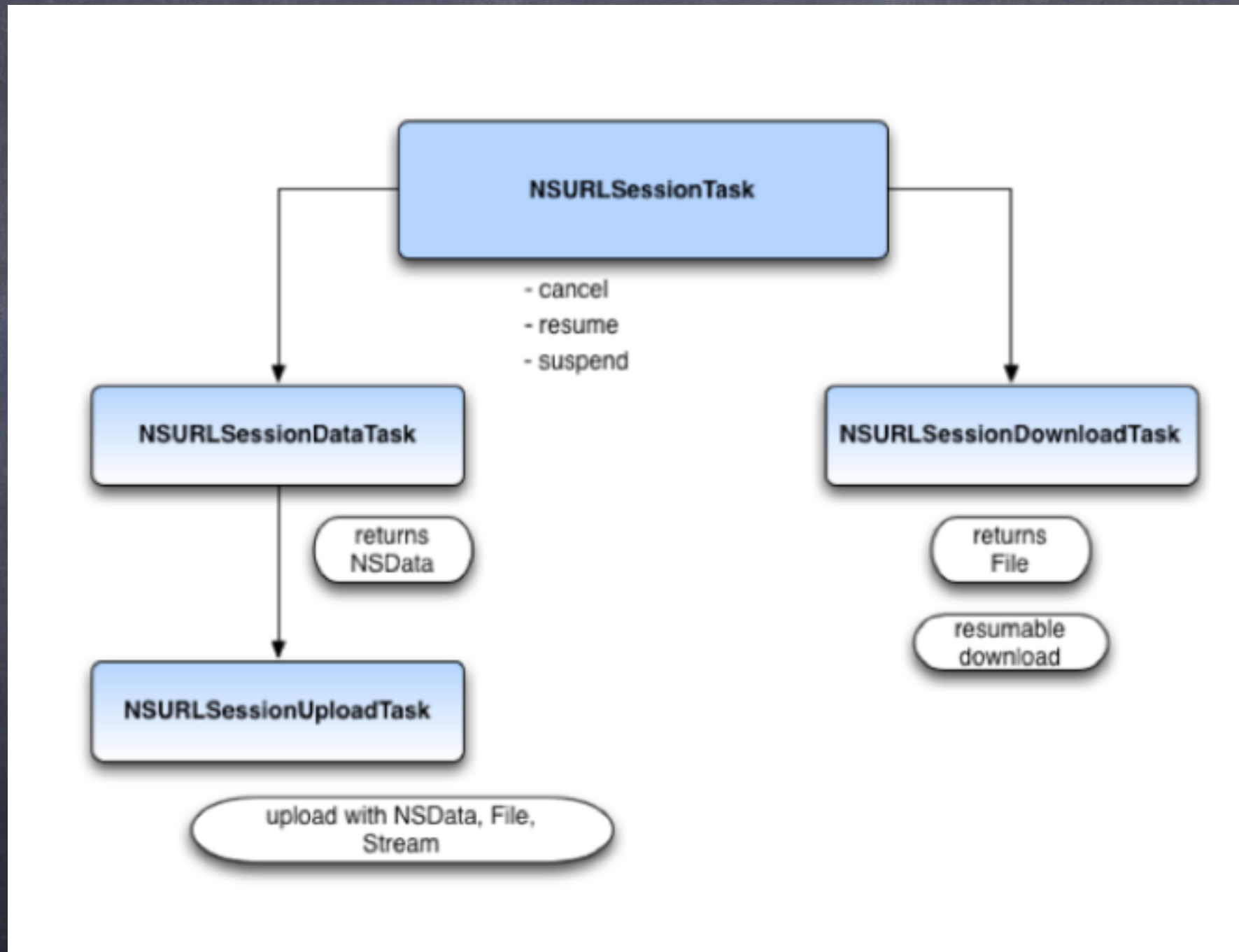

NSURLSessionTask

- Uma vez que a sessão é criada e configurada, pode se criar tarefas para serem executadas como download de arquivos e upload de dados. Tarefas são criadas através da classe NSURLSessionTask. Existem três tipos de tarefas possíveis:
 - NSURLSessionDataTask: criam requisições e recebem respostas dos servidores, com dados do tipo NSData em memória
 - NSURLSessionUploadTask: facilitam o processo de upload de dados
 - NSURLSessionDownloadTask: efetuam download de dados diretamente em disco, diferente da NSURLSessionDataTask que recebe os dados em memória



NSURLSessionTask

• Outras Características das Tasks



NSURLSessionTask

- Uma das maneiras de se criar uma task (NSURLSessionDataTask) é através do método `dataTaskWithURL` da `NSURLSession`. Esse método pode ser pensando em algo como solicitar ao app que: "execute a tarefa de requisição nessa URL e guarde os dados".
- Faça a programação das linhas abaixo, a linha 30 já foi feita em slide anterior.

```
28
29 //URL de acesso a API do itunes para retornar o App top Free
30 let url = NSURL (string:"https://itunes.apple.com/br/rss/topfreeapplications/limit=1/json")
31
32 let task = session!.dataTaskWithURL(url!,
33                                     completionHandler: {
34                                         (data: NSData?, response:NSURLResponse?, error: NSError?) -> Void in
35
36                                         //ações que serão efetuadas quando a execução da task se completa
37
38                                     })
39 //disparo da execução da task
40 task.resume()
41 }
```

- Este método cria uma task usando requisitando dados na URL definida. Ao completar (completionHandler) permite acesso aos dados de retorno através de uma closure, que é espécie de um bloco de execução (função inline). Nesse caso esse bloco da tarefa possui três parâmetros:
1. **data:** NSData: buffer que recebe os dados em bytes. Precisa ser convertido para ser trabalhado em outros formatos

NSURLSessionTask

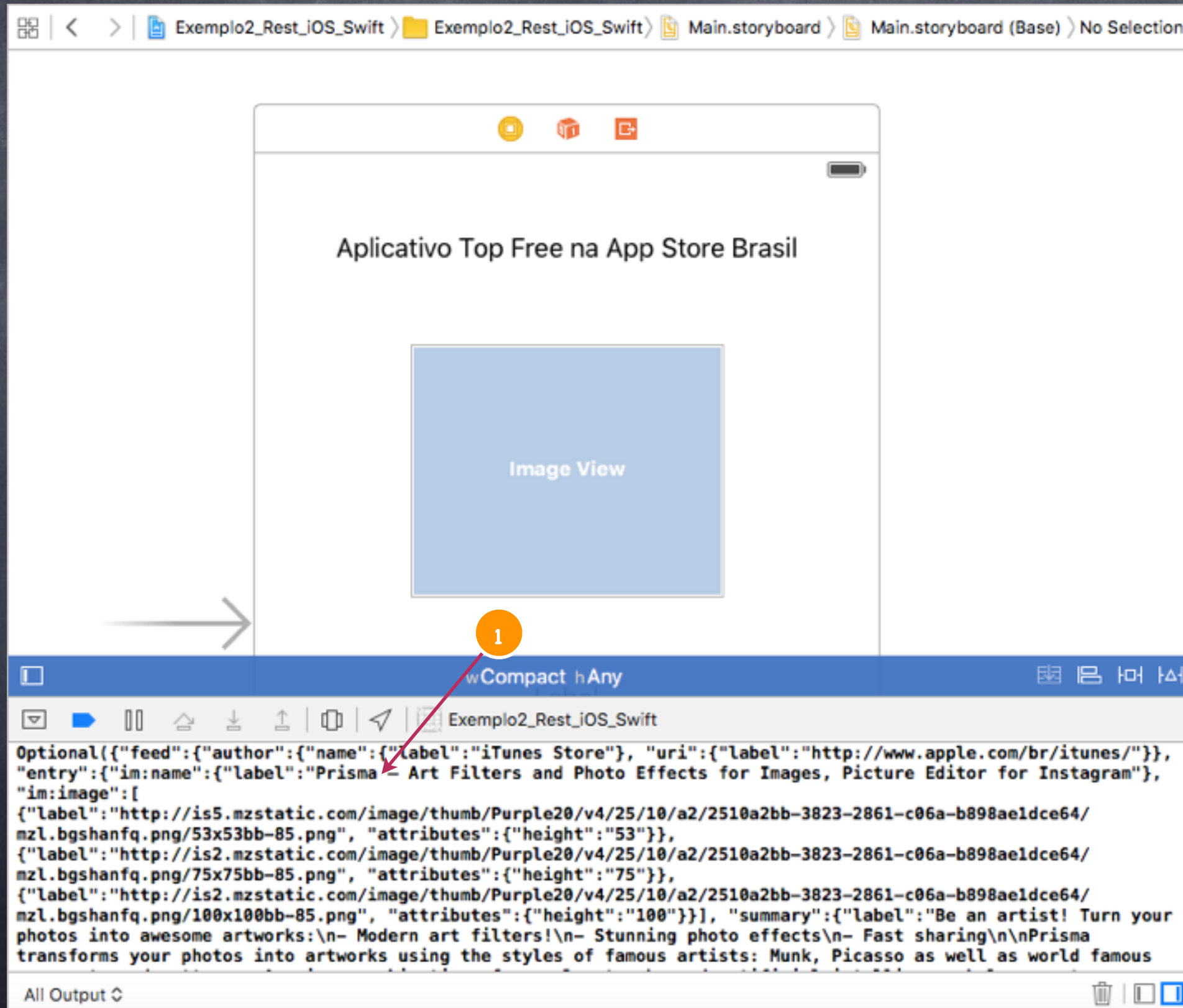
2. **response:** NSURLResponse: uma resposta onde é possível checar por exemplo se ocorreu um erro através do status do HTTP
3. **error:** NSError: outros tipos de erro

- Implemente as linhas 37 e 38 para exibir o Json:

```
29 //URL de acesso a API do itunes para retornar o App top Free
30 let url = NSURL (string:"https://itunes.apple.com/br/rss/topfreeapplications/limit=1/json")
31
32 let task = session!.dataTaskWithURL(url!,
33                                     completionHandler: {
34                                         (data: NSData?, response:NSURLResponse?, error: NSError?) -> Void in
35
36                                         //ações que serão efetuadas quando a execução da task se completa
37                                         let string = NSString(data: data!, encoding: NSUTF8StringEncoding)
38                                         print(string)
39                                     })
40 //disparo da execução da task
41 task.resume()
42 }
43
```


Executando o App

- Ao executar o programa, irá aparecer no console o conteúdo do Json (1).



Estrutura do Json da App Store

```
2016-07-11 13:05:08.974 Exemplo2_Rest_iOS[3160:186332] {
  feed = {
    author = {
      name = {
        label = "iTunes Store";
      };
      uri = {
        label = "http://www.apple.com/br/itunes/";
      };
    };
    entry = {
      category = {
        attributes = {
          "im:id" = 6003;
          label = Viagens;
          scheme = "https://itunes.apple.com/br/genre/ios-viagens/id6003?mt=8&uo=2";
          term = Travel;
        };
      };
      id = {
        attributes = {
          "im:bundleId" = "com.ubercab.UberClient";
          "im:id" = 368677368;
        };
        label = "https://itunes.apple.com/br/app/uber/id368677368?mt=8&uo=2";
      };
      "im:artist" = {
        attributes = {
          href = "https://itunes.apple.com/br/developer/uber-technologies-inc./id368677371?mt=8&uo=2";
        };
        label = "Uber Technologies, Inc.";
      };
      "im:contentType" = {
        attributes = {
          label = Aplicativo;
          term = Application;
        };
      };
      "im:image" = (
        [0] {
          attributes = {
            height = 53;
          };
          label = "http://is2.mzstatic.com/image/thumb/Purple18/v4/d3/62/52/d36252b8-4d49-cf57-df8f-6a03fc18e520/mzl.vtoredrs.png/53x53bb-85.png";
        },
        [1] {
          attributes = {
            height = 75;
          };
          label = "http://is4.mzstatic.com/image/thumb/Purple18/v4/d3/62/52/d36252b8-4d49-cf57-df8f-6a03fc18e520/mzl.vtoredrs.png/75x75bb-85.png";
        }
      )
    }
  }
}
```


Continuação

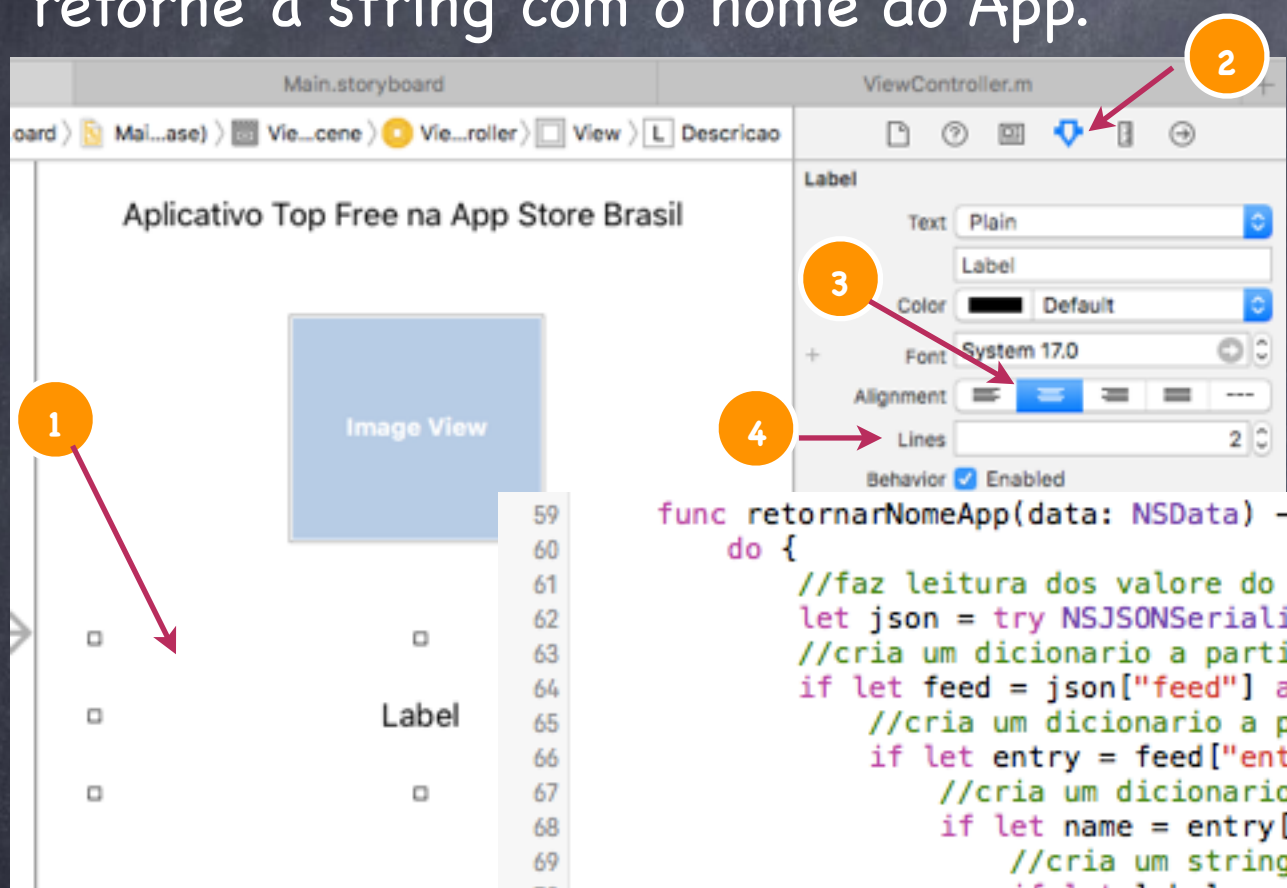
```
    },
    {
      [2] attributes =
            {
              height = 100;
            };
      label = "http://is4.mzstatic.com/image/thumb/Purple18/v4/d3/62/52/d36252b8-4d49-cf57-df8f-6a03fc18e520/mzl.vtoredrs.png/100x100bb-85.png";
    }
  );
  "im:name" =
    {
      label = Uber;
    };
  "im:price" =
    {
      attributes =
        {
          amount = "0.00000";
          currency = USD;
        };
      label = 0bter;
    };
  "im:releaseDate" =
    {
      attributes =
        {
          label = "20/05/2010";
        };
      label = "2010-05-20T20:11:23-07:00";
    };
  link =
    {
      attributes =
        {
          href = "https://itunes.apple.com/br/app/uber/id368677368?mt=8&uo=2";
          rel = alternate;
          type = "text/html";
        };
    };
  rights =
    {
      label = "\U00a9 Uber Technologies Inc.";
    };
  summary =
    {
      label = "Consiga uma viagem confi\U00elvel em minutos com o aplicativo Uber \U2014 sem reservas ou filas de espera de t\U00elxi. \n
\nTodas as op\U00e7\U00f5es, desde a viagem de baixo custo at\U00e9 a premium, funcionam como uma atualiza\U00e7\U00e3o aos sistemas comuns. \n\nCrie sua conta para explorar o aplicativo. Adicione um cart\U00e3o de cr\U00e9dito ou vincule seu PayPal, que sua tarifa ser\U00el cobrada automaticamente no final da sua viagem. Voc\U00ea tamb\U00e9m pode pagar com dinheiro em algumas cidades. Depois de sua viagem, vamos enviar um recibo a voc\U00ea por e-mail. \n\nVeja se o Uber est\U00el dispon\U00edvel em sua cidade em https://www.uber.com/cities\nSiga-nos no Twitter em https://twitter.com/uber\nCurta-nos no Facebook em https://www.facebook.com/uber\n\nD\U00favidas? Acesse https://help.uber.com/";
    };
};
```


Continuação

```
        label = "\U00a9 Uber Technologies Inc.";
    };
    → summary =
        {
            label = "Consiga uma viagem confi\U00elvel em minutos com o aplicativo Uber \U2014 sem reservas ou filas de espera de t\U00elxi. \n
\nTodas as op\U00e7\U00f5es, desde a viagem de baixo custo at\U00e9 a premium, funcionam como uma atualiza\U00e7\U00e3o aos sistemas comuns. \n\nCrie
sua conta para explorar o aplicativo. Adicione um cart\U00e3o de cr\U00e9dito ou vincule seu PayPal, que sua tarifa ser\U00el cobrada automaticamente no
final da sua viagem. Voc\U00ea tamb\U00e9m pode pagar com dinheiro em algumas cidades. Depois de sua viagem, vamos enviar um recibo a voc\U00ea por e-
mail. \n\nVeja se o Uber est\U00el dispon\U00edvel em sua cidade em https://www.uber.com/cities\nSiga-nos no Twitter em https://twitter.com/uber\nCurta-
nos no Facebook em https://www.facebook.com/uber\n\nD\U00favidas? Acesse https://help.uber.com/";
        };
    → title =
        {
            label = "Uber - Uber Technologies, Inc.";
        };
    };
    → icon =
        {
            label = "http://itunes.apple.com/favicon.ico";
        };
    → id =
        {
            label = "https://itunes.apple.com/br/rss/topfreeapplications/limit=1/json";
        };
    → link =
        (
            {
                attributes =
                    {
                        href = "https://itunes.apple.com/WebObjects/MZStore.woa/wa/viewTop?cc=br&id=132006&popId=27";
                        rel = alternate;
                        type = "text/html";
                    };
            },
            {
                attributes =
                    {
                        href = "https://itunes.apple.com/br/rss/topfreeapplications/limit=1/json";
                        rel = self;
                    };
            }
        );
    → rights =
        {
            label = "Copyright 2008 Apple Inc.";
        };
    → title =
        {
            label = "iTunes Store: Top apps gr\U00eltis";
        };
    → updated =
        {
            label = "2016-07-11T08:39:55-07:00";
        };
    };
}
```


Exibir o nome Top Free

- Selecione e aumente a área do Label (1), em propriedades (2), alinhe o texto ao centro (3) e deixe formatado para o texto aparecer em 2 linhas (4). Para que seja possível penetrar na estrutura do Json e acessar o nome do aplicativo, crie uma função chamada **retornarNomeApp** que receba o NSData, avance pelos dicionários e retorne a string com o nome do App.



The screenshot shows the Xcode interface with two panes. The left pane (Main.storyboard) displays a storyboard with a blue box labeled 'Image View' and a 'Label' below it. An orange circle with the number '1' points to the 'Label' in the storyboard. The right pane (ViewController.m) shows the code editor. An orange circle with the number '2' points to the 'Properties' panel on the right side of the code editor. An orange circle with the number '3' points to the 'Alignment' property in the 'Label' properties panel. An orange circle with the number '4' points to the 'Lines' property in the 'Label' properties panel, which is set to '2'.

```
59 func retornarNomeApp(data: NSData) -> String? {
60     do {
61         //faz leitura dos valores do JSON, NSJSONSerialization faz o Parser do Json
62         let json = try NSJSONSerialization.JSONObjectWithData(data, options: []) as! [String: AnyObject]
63         //cria um dicionário a partir da chave "feed"
64         if let feed = json["feed"] as? [String:AnyObject] {
65             //cria um dicionário a partir da chave "entry"
66             if let entry = feed["entry"] as? [String:AnyObject] {
67                 //cria um dicionário a partir da chave "im:name"
68                 if let name = entry["im:name"] as? [String:AnyObject]{
69                     //cria um string a partir da chave "label"
70                     if let label = name["label"] as? String{
71                         return label
72                     }
73                 }
74             }
75         }
76     } catch let error as NSError {
77         return "Falha ao carregar : \(error.localizedDescription)"
78     }
79     return nil
80 }
81
82
```

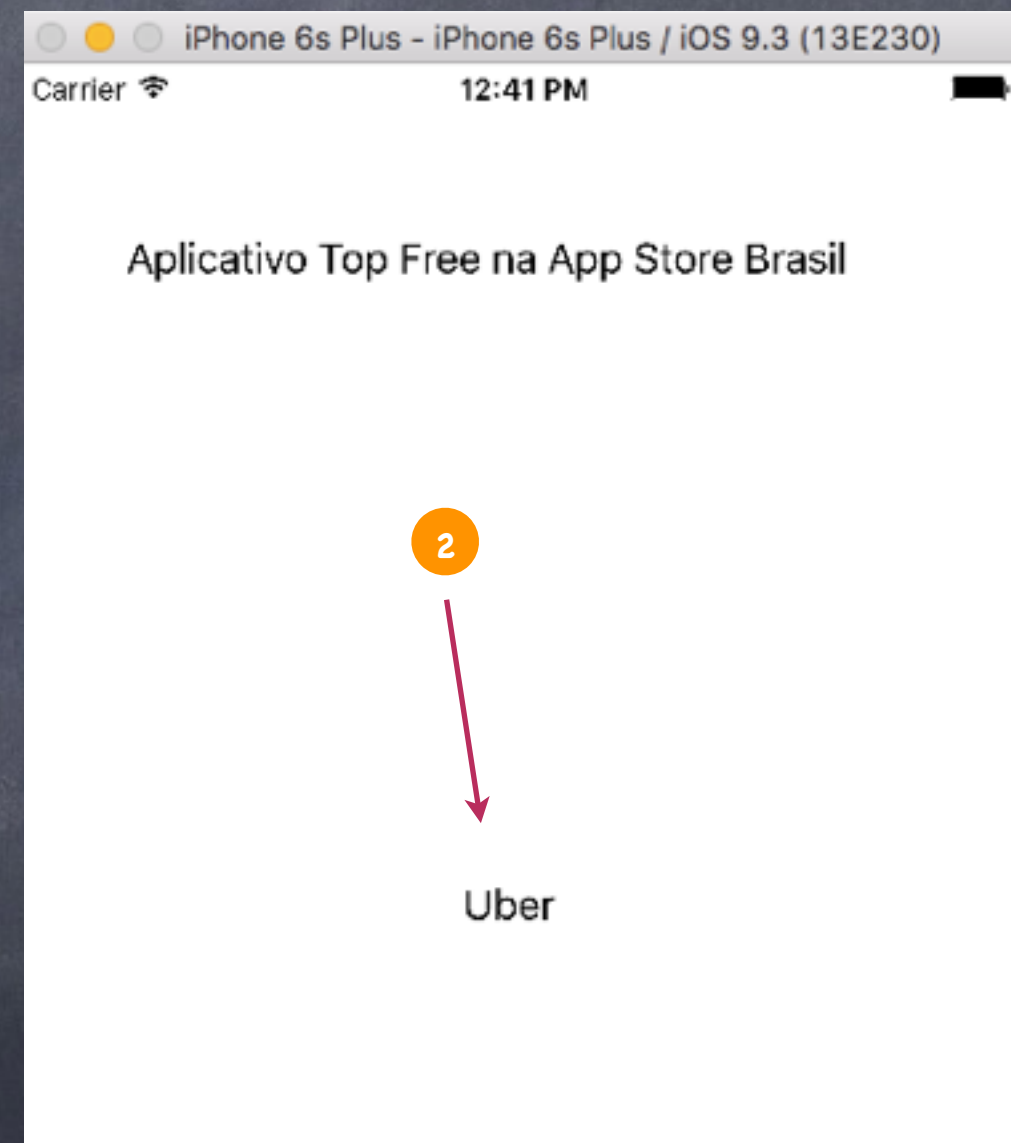
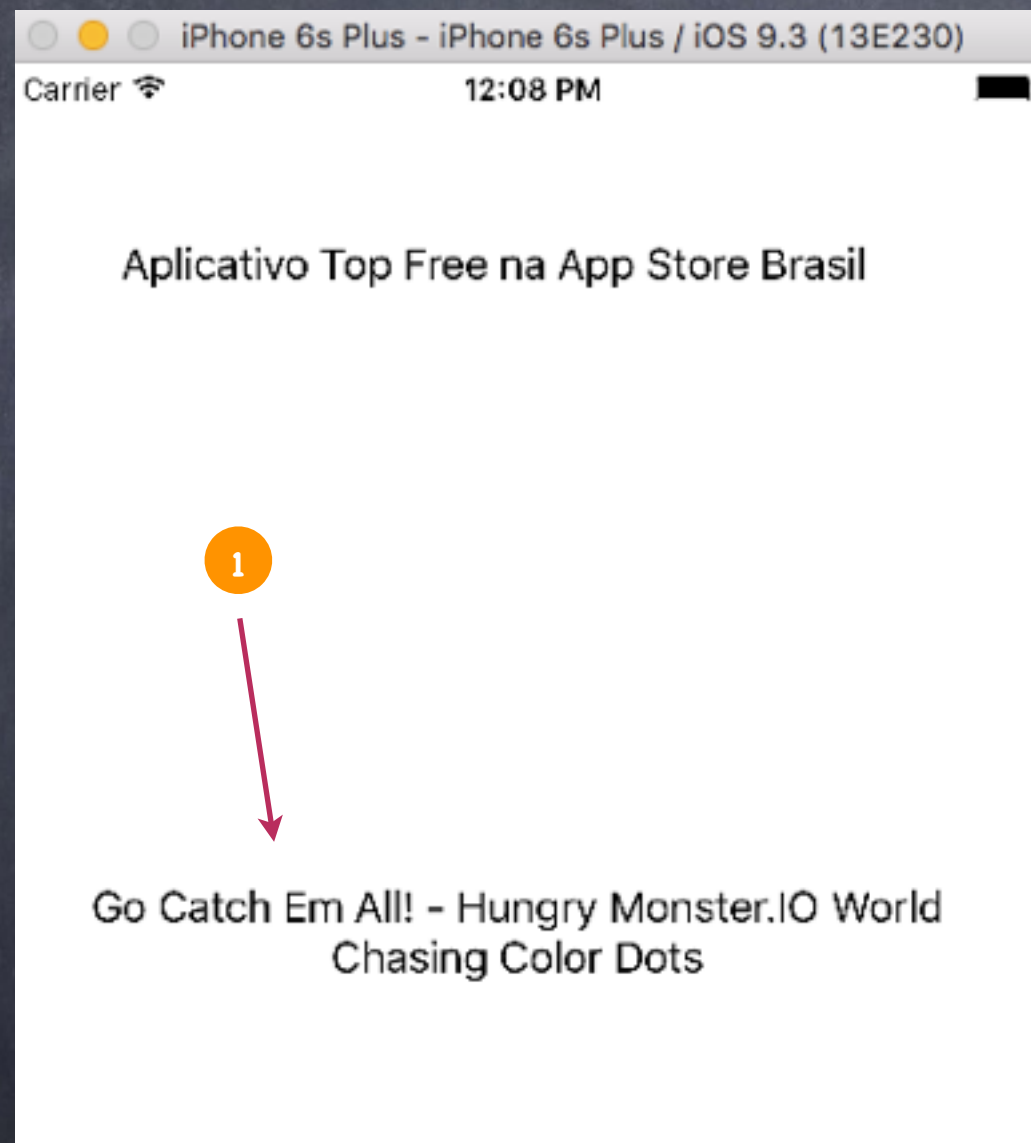

Exibir o nome Top Free

- Depois de criar a função retornarNomeApp, implemente a linha 40 para descarregar o nome que retorna da função para a constante nomeApp, quando utilizamos o dataTaskWithURL a tarefa não é executada na thread principal do App, utilizando a programação da linha 41 até a 43 o label será atualizado de forma mais rápida na thread principal.

```
31
32     let task = session!.dataTaskWithURL(url!,
33                                         completionHandler: {
34                                             (data: NSData?, response: NSURLConnection?, error: NSError?) -> Void in
35
36                                             //ações que serão executadas qdo a execução da task é completada
37                                             let string = NSString(data: data!, encoding: NSUTF8StringEncoding)
38                                             print(string)
39
40                                             if let nomeApp = self.retornarNomeApp(data!){
41                                                 dispatch_async(dispatch_get_main_queue(), {
42                                                     self.descricao.text = nomeApp
43                                                 })
44                                             }
45                                         })
46     //disparo da execução da task
47     task.resume()
48 }
49
```


Resultado

- Ao executar o programa irá aparecer o nome do Top Free da Apple Store no dia 11/07/2016, exemplo de dois momentos às 12h08 (1) e às 12:41 (2)



Exibindo a imagem

- As imagens ficam em um array dentro de um dicionário chamado "im:image", implemente uma função que retorne uma string com a URL da imagem. Veja na imagem a estrutura do Array no Json, observe que existem 3 tamanhos de imagem, uma em cada índice, o índice 0 tem a imagem com altura 53, o índice 1 tem a imagem com altura 75 e o índice 2 tem a imagem com altura 100, acesse a url da imagem escolhendo o índice apropriado (0,1 ou 2) e utilize o dicionário com a chave "label" para acessar a URL da imagem.

1

```
2016-07-11 12:39:58.006 Exemplo2_Rest_iOS[2925:171158] (  
  {  
    attributes = {  
      height = 53; [0]  
    };  
    label = "http://is2.mzstatic.com/image/thumb/Purple18/v4/d3/62/52/d36252b8-4d49-cf57-df8f-6a03fc18e520/mzl.vtoredrs.png/53x53bb-85.png";  
  },  
  {  
    attributes = {  
      height = 75; [1]  
    };  
    label = "http://is4.mzstatic.com/image/thumb/Purple18/v4/d3/62/52/d36252b8-4d49-cf57-df8f-6a03fc18e520/mzl.vtoredrs.png/75x75bb-85.png";  
  },  
  {  
    attributes = {  
      height = 100; [2]  
    };  
    label = "http://is4.mzstatic.com/image/thumb/Purple18/v4/d3/62/52/d36252b8-4d49-cf57-df8f-6a03fc18e520/mzl.vtoredrs.png/100x100bb-85.png";  
  }  
)
```


Exibindo a imagem

- Implemente uma função que retorne uma string com a URL da imagem, as imagens ficam em um array dentro de um dicionário chamado "im:image", penetre na estrutura do Json pelos dicionários até chegar ao array e acesse o índice [2] para acessar a URL da maior imagem disponível.

```
83 func retornarImagemApp(data: NSData) -> String? {
84     do {
85         //faz leitura dos valores do JSON, NSJSONSerialization faz o Parser do Json
86         let json = try NSJSONSerialization.JSONObjectWithData(data, options: []) as! [String: AnyObject]
87         //cria um dicionário a partir da chave "feed"
88         if let feed = json["feed"] as? [String:AnyObject] {
89             //cria um dicionário a partir da chave "entry"
90             if let entry = feed["entry"] as? [String:AnyObject] {
91                 //cria um dicionário a partir da chave "im:image"
92                 if let name = entry["im:image"] as? [AnyObject]{
93                     // cria uma string para receber a url da imagem na terceira posição do array
94                     if let image = name[2] as? [String: AnyObject]{
95                         //cria um string a partir da chave "label"
96                         if let label = image["label"] as? String{
97                             return label
98                         }
99                     }
100                 }
101             }
102         }
103     } catch let error as NSError {
104         return "Falha ao carregar : \(error.localizedDescription)"
105     }
106     return nil
107 }
108 }
```


Exibindo a imagem

- Implemente uma nova função chamada `carregarImagemdeURL`, essa função receberá a string com a URL e carregará no Outlet a imagem vinda pela URL. Para se recuperar imagens é preciso efetuar o download das mesmas em bytes e os transformar em um objeto do tipo `UIImage` para ser exibido na interface.

```
112 func carregarImagemdeURL(imageURL : String) {
113     // Cria uma Url de uma string com a URL
114     let url = NSURL(string: imageURL)!
115     // Cria uma taks do tipo Download
116     // - sharedInstance = global NSURLCache, NSHTTPCookieStorage and NSURLConnection objects.
117     let task = NSURLSession.sharedSession().dataTaskWithURL(url) { (responseData, responseUrl, error) -> Void in
118         // se responseData is not null...
119         // recebe o binário da imagem
120         if let imageData = responseData{
121             //transforma o binario em UIImage e atualiza a tela na thread principal
122             dispatch_async(dispatch_get_main_queue(), { () -> Void in
123                 self.imagem.image = UIImage(data: imageData)
124             })
125         }
126     }
127     //disparo da execução da task
128     task.resume()
129 }
```

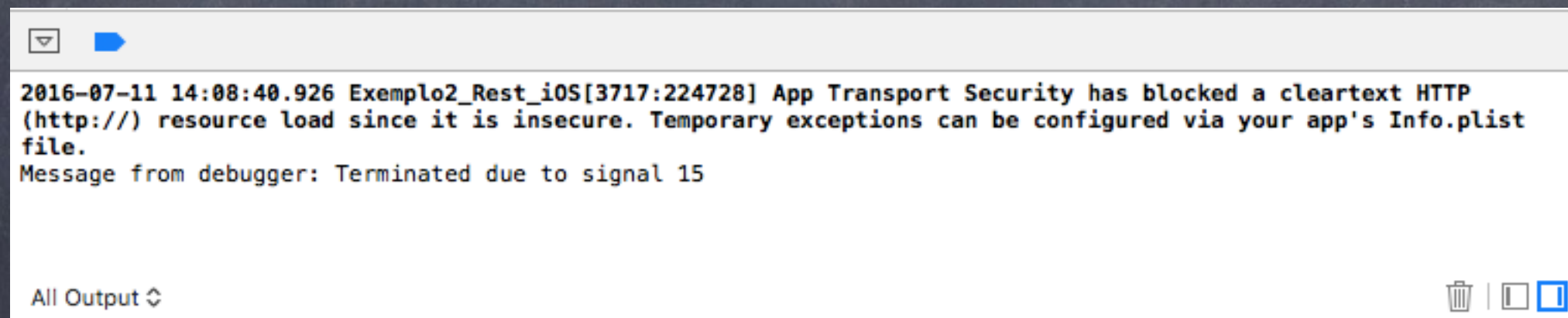

Exibindo a imagem

- O último passo é chamar a função que acessa a URL da imagem e faz o carregamento dessa url convertendo para um objeto do tipo UIImage. Essas ações são feitas nas linhas 46 a 48.

```
32     let task = session!.dataTaskWithURL(url!,
33                                     completionHandler: {
34                                         (data: NSData?, response: NSURLResponse?, error: NSError?) -> Void in
35
36                                             //ações que serão executadas qdo a execução da task é completada
37                                             //let string = NSString(data: data!, encoding: NSUTF8StringEncoding)
38                                             //print(string)
39
40                                             if let nomeApp = self.retornarNomeApp(data!){
41                                                 dispatch_async(dispatch_get_main_queue(), {
42                                                     self.descricao.text = nomeApp
43                                                 })
44                                             }
45
46                                             if let appImageURL = self.retornarImagemApp(data!){
47                                                 self.carregarImagemdeURL(appImageURL)
48                                             }
49                                     })
50     //disparo da execução da task
51     task.resume()
52 }
```


Exibindo a imagem

- Ao executar observe o erro que ocorre quando você tenta exibir uma imagem utilizando o http ao invés de https, sobre essas informações de segurança reveja o slide aula_13_2016_WebView_Swift página 18 até a 21.



```
2016-07-11 14:08:40.926 Exemplo2_Rest_iOS[3717:224728] App Transport Security has blocked a cleartext HTTP (http://) resource load since it is insecure. Temporary exceptions can be configured via your app's Info.plist file.  
Message from debugger: Terminated due to signal 15
```

All Output ↕

- Para continuar o exercício veja na próxima página uma das soluções que está disponível na página 21 do slide aula_13_2016_WebView_ObjC.

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 Loads (5), selecione YES (6).

The screenshot illustrates the steps to configure App Transport Security Settings in the Info.plist file. The interface is divided into three main sections: the Project Navigator on the left, the Properties Inspector in the center, and the Attributes Inspector on the right.

Project Navigator (Left): Shows the project structure for 'Exemplo2_Rest_iOS_Swift'. The 'Info.plist' file is highlighted with a red arrow and a yellow circle labeled '1'.

Properties Inspector (Center): Displays the 'Information Property List' for the selected file. The 'Application Category' key is selected at the bottom, with a red arrow and a yellow circle labeled '2' pointing to the '+' button next to it.

Attributes Inspector (Right): Shows the 'App Transport Security Settings' dictionary. A red arrow and a yellow circle labeled '3' point to the 'App Transport Security Settings' option in the list. Below this, a red arrow and a yellow circle labeled '4' point to the triangle icon next to the 'App Transport Security Settings' header, indicating it can be expanded.

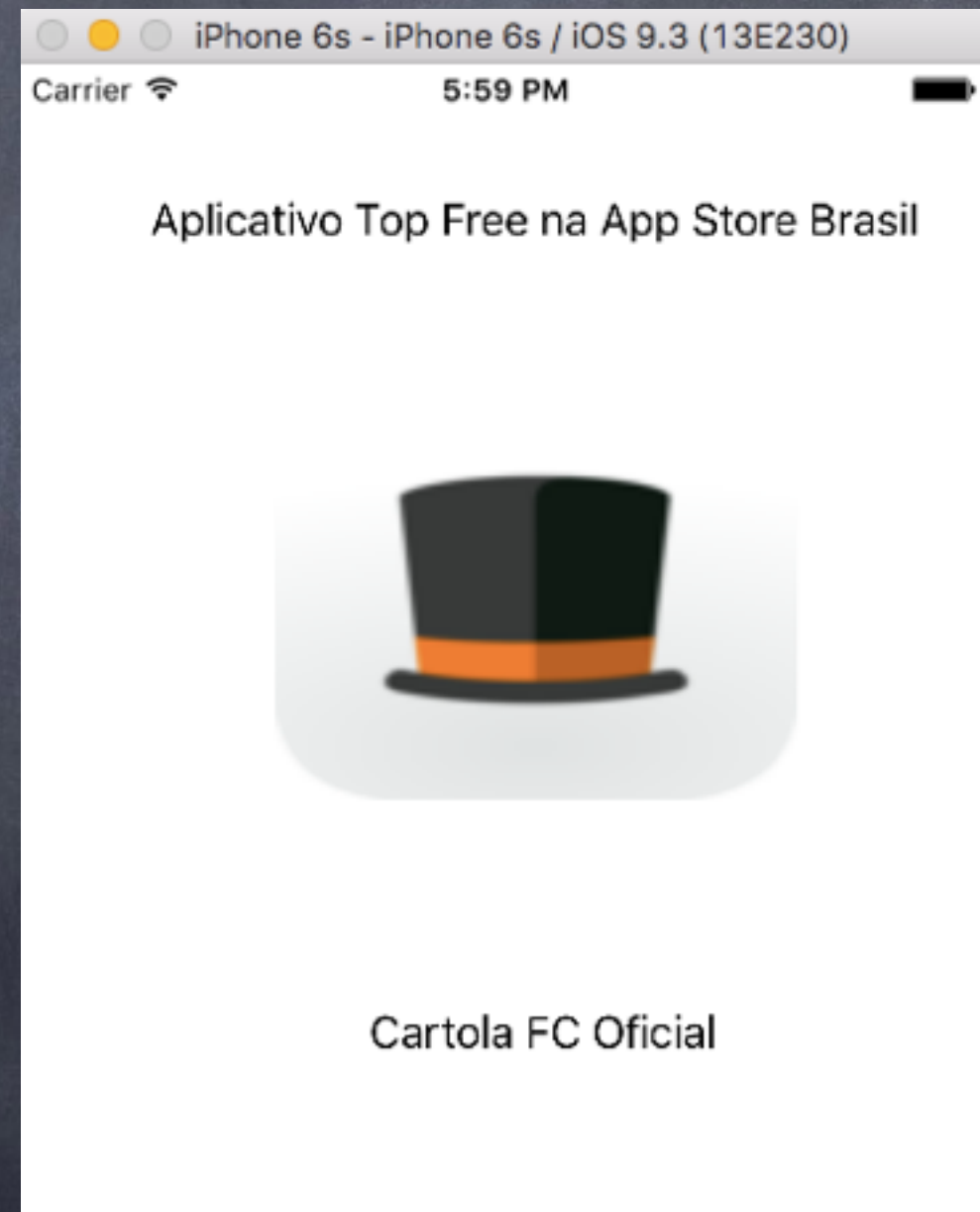
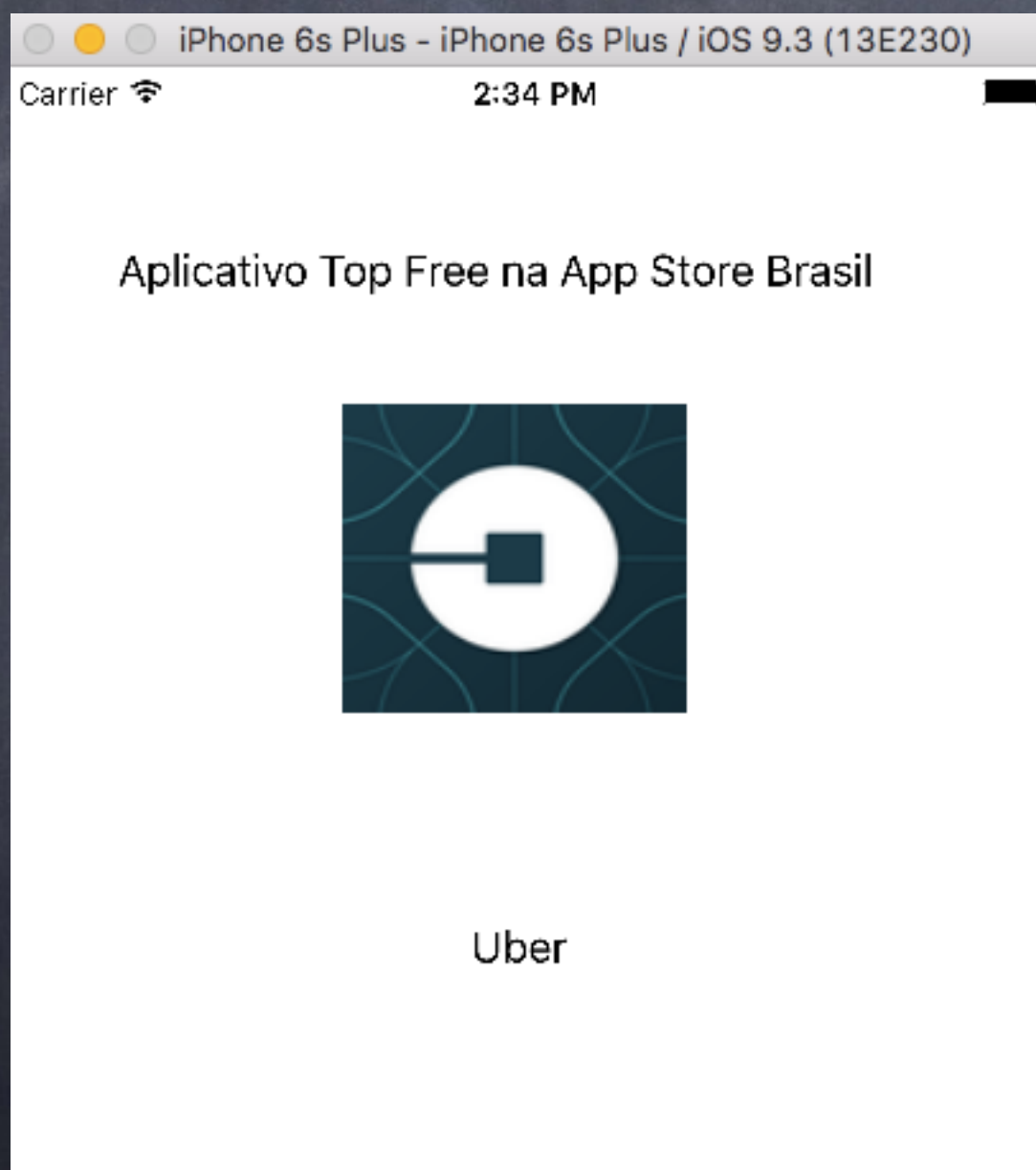
Expanded Dictionary (Bottom): Shows the expanded 'App Transport Security Settings' dictionary. A red arrow and a yellow circle labeled '5' point to the 'Allow Arbitrary Loads' key. Another red arrow and a yellow circle labeled '6' point to the 'YES' value for this key.

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

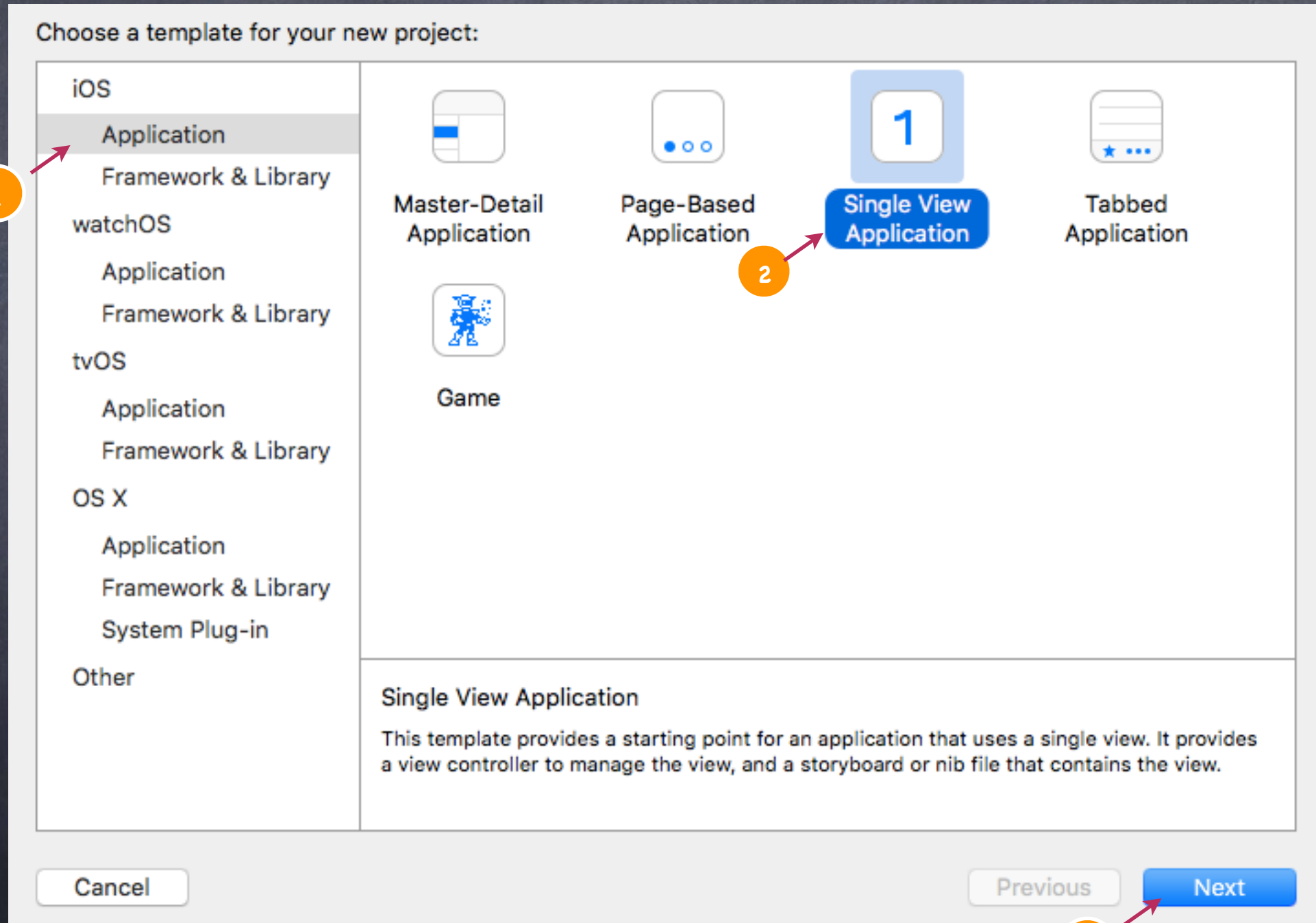
Exibindo a imagem

- Observe as imagens dos App's top Free em dois momentos, a primeira imagem é do dia 11/07/16 às 2:34 PM, a segunda imagem é do dia 12/07/16 às 5:59 PM



Iniciando outro Projeto

- Clique em File -> New Project -> iOS -> Application -> Single View Application.



Os dados do Projeto

- Preencha com os dados abaixo, lembre-se que o Organization Identifier é como se fosse o pacote no Java ou o namespace do VB, em language escolha Swift, em Devices selecione iPhone.

Choose options for your new project:

Product Name:

4 Organization Name:

Organization Identifier:

Bundle Identifier:

Language:

5 Devices:

6

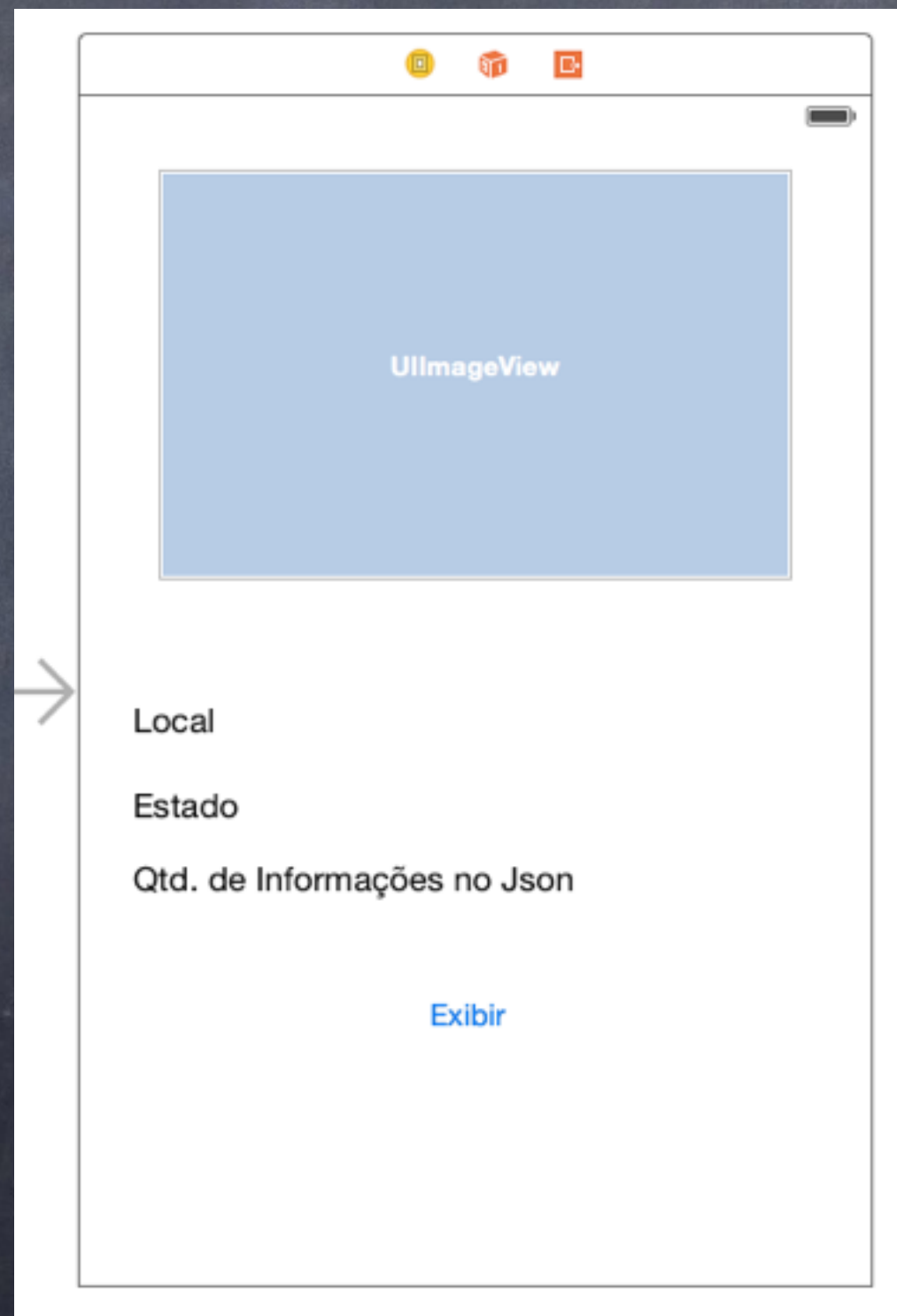
☐ Use Core Data

☐ Include Unit Tests

☐ Include UI Tests

A Interface

- Desenhe a tela abaixo com 3 label's, 1 button e 1 UIImageView.



Definindo os IBOutlet's

- Abra duas telas simultâneas e vamos criar no .swift os Outlet's dos Labels e do Image. Com botão direito sobre o objeto já selecionado (1), escolha New Referencing Outlet e arraste dentro da área das chaves (2). Para o Image nomeie como imagem, para os Label's use respectivamente os nomes local, estado e qtdInfo.

Local

Estado

Qtd. de Informações no Json

Exibir

```

1 //
2 // ViewController.swift
3 // Exemplo1_Rest_iOS_Swift
4 //
5 // Created by agesandro scarpioni on 12/07/16.
6 // Copyright © 2016 Agesandro Scarpioni. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController {
12
13
14
15
16
17     override func viewDidLoad() {
18         super.viewDidLoad()
19         // Do any additional setup after loading the view, typical
20     }
21
22     override func didReceiveMemoryWarning() {
23         super.didReceiveMemoryWarning()
24         // Dispose of any resources that can be recreated.
25     }
26
27 }
28
29
30

```


Definindo os IBAction's

- Para o botão crie um Action (1), nomeie como exibir e descarregue na área indicada (2)

The screenshot shows the Xcode interface with a storyboard on the left and a Swift file on the right. A dialog box for creating an IBAction is open, with the following settings:

- Connection: Action (indicated by a red arrow and a yellow circle with the number 1)
- Object: View Controller
- Name: exibir
- Type: AnyObject
- Event: Touch Up Inside
- Arguments: Sender

Below the dialog, the storyboard elements are visible: a blue rectangle labeled "Imagem", a label "Estado", and a button labeled "Exibir" (indicated by a red arrow).

On the right, the Swift code for ViewController.swift is shown. A red arrow points to the `viewDidLoad()` method, which is the area where the action should be connected (indicated by a yellow circle with the number 2).

```

1 //
2 // ViewController.swift
3 // Exemplo1_Rest_iOS_Swift
4 //
5 // Created by agesandro scarpioni on 12/07/16.
6 // Copyright © 2016 Agesandro Scarpioni. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController {
12
13     @IBOutlet weak var Imagem: UIImageView!
14     @IBOutlet weak var local: UILabel!
15     @IBOutlet weak var estado: UILabel!
16     @IBOutlet weak var qtdInfo: UILabel!
17
18
19
20     override func viewDidLoad() {
21         super.viewDidLoad()
22         // Do any additional setup after loading the view, typical
23     }
24
25     override func didReceiveMemoryWarning() {
26         super.didReceiveMemoryWarning()
27         // Dispose of any resources that can be recreated.
28     }
29
30 }
31
32
33
34

```


Implementando o Exibir

- Nas linhas abaixo será criada uma sessão e a configuração da sessão para realizar o request e posteriormente o parser do Json para um NSDictionary, a url scarpioni.com é de um webservice em php hospedado no meu servidor GoDaddy acessando Mysql, a outra url é de um webservice em Node.JS hospedado no Heroku acessando banco MongoDB.

```
1 //
2 // ViewController.swift
3 // Exemplo1_Rest_iOS_Swift
4 //
5 // Created by agesandro scarpioni on 12/07/16.
6 // Copyright © 2016 Agesandro Scarpioni. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController {
12
13     @IBOutlet weak var Imagem: UIImageView!
14     @IBOutlet weak var local: UILabel!
15     @IBOutlet weak var estado: UILabel!
16     @IBOutlet weak var qtdInfo: UILabel!
17
18     var session: NSURLSession?
19
20     @IBAction func exibir(sender: AnyObject) {
21         //cria um configuracao de sessao default
22         let sessionConfig = NSURLSessionConfiguration.defaultSessionConfiguration()
23
24         //cria uma sessao com a configuracao default
25         session = NSURLSession(configuration: sessionConfig)
26
27         //acesso a API feita em PHP criada em meu servidor na Amazon
28         //let url = NSURL (string:"https://www.scarpioni.com/webservices/local.php?id=1")
29
30         let url = NSURL (string:"https://parks-api.herokuapp.com/parks/577024e4a44821110001ee93")
31     }
```


NSURLSessionTask

- Uma vez que a sessão é criada e configurada, pode se criar tarefas para serem executadas como download de arquivos e upload de dados. Tarefas são criadas através da classe NSURLSessionTask, implemente as linhas 32 até a 41

```
28 //acesso a API feita em PHP criada em meu servidor na Godaddy
29 //let url = NSURL (string:"https://www.scarpioni.com/webservices/local.php/?id=1")
30 //acesso a API feita em Node.JS criada no heroku
31 let url = NSURL (string:"https://parks-api.herokuapp.com/parks/577024e4a44821110001ee93")
32 //fazendo o request da url em uma task
33 let task = session!.dataTaskWithURL(url!,
34                                     completionHandler: {
35                                         (data: NSData?, response:NSURLResponse?, error: NSError?) -> Void in
36
37                                         //ações que serão executadas qdo a execução da task é completada
38                                         let string = NSString(data: data!, encoding: NSUTF8StringEncoding)
39                                         print(string)
40                                     })
41 //disparo da execução da task
42 task.resume()
43 }
```

- Ao executar o App e clicar no botão Exibir, os dados do Json irá aparecer no console como mostra a figura abaixo:

```
Optional({"_id":"577024e4a44821110001ee93","usuario":"5779ae8c59f2471f13651ede","nome":"Parque Villa Lobos","cidade":"São Paulo","estado":"SP","urlfoto":"http://msalx.vejasp.abril.com.br/2012/10/08/1103/jcclk/parque-villa-lobos-3.jpeg","datacriacao":"2016-06-26T18:54:28.872Z","__v":0})
```

All Output ↕



Implementando o Exibir

- Crie a variável qtd no topo da classe (1), faça um teste apenas para armazenar a quantidade de chaves que existe no json (id, usuario, nome, cidade, estado, urlfoto, datacriacao, _v) totalizando 8.

```

9  import UIKit
10
11  class ViewController: UIViewController {
12
13      @IBOutlet weak var Imagem: UIImageView!
14      @IBOutlet weak var local: UILabel!
15      @IBOutlet weak var estado: UILabel!
16      @IBOutlet weak var qtdInfo: UILabel!
17
18      var session: URLSession?
19      var qtd = 0
20

```

- Crie uma função que receba o NSData para que seja feito o parser, e por meio do dicionário com chave "nome" seja possível retornar o nome do parque.

```

68  func retornarNomePq(data: NSData) -> String? {
69      var resposta:String?=nil
70      do {
71          //faz leitura dos valores do JSON, NSJSONSerialization faz o Parser do Json
72          let json = try NSJSONSerialization.JSONObjectWithData(data, options: []) as! [String: AnyObject]
73          //cria uma string a partir da chave "nome"
74          if let nomeParque = json["nome"] as? String {
75              resposta = nomeParque
76              qtd = json.count //apenas um exemplo para contar a qtd de chaves no json
77          }
78      } catch let error as NSError {
79          resposta = "Falha ao carregar : \(error.localizedDescription)"
80      }
81      return resposta
82  }

```


Implementando o Exibir

- Crie a variável qtd no topo da classe (1) apenas para armazenar a quantidade de chaves que existe no json (id, usuario, nome, cidade, estado, urlfoto, datacriacao, _v) totalizando 8.

```

9  import UIKit
10
11  class ViewController: UIViewController {
12
13      @IBOutlet weak var Imagem: UIImageView!
14      @IBOutlet weak var local: UILabel!
15      @IBOutlet weak var estado: UILabel!
16      @IBOutlet weak var qtdInfo: UILabel!
17
18      var session: NSURLSession?
19      var qtd = 0 ← 1
20

```

- Crie uma função que receba o NSData para que seja feito o parser, e por meio do dicionário com chave "nome" seja possível retornar o nome do parque.

```

68  func retornarNomePq(data: NSData) -> String? {
69      var resposta:String?=nil
70      do {
71          //faz leitura dos valore do JSON, NSJSONSerialization faz o Parser do Json
72          let json = try NSJSONSerialization.JSONObjectWithData(data, options: []) as! [String: AnyObject]
73          //cria uma string a partir da chave "nome"
74          if let nomeParque = json["nome"] as? String {
75              resposta = nomeParque
76              qtd = json.count //apenas um exemplo para contar a qtd de chaves no json
77          }
78      } catch let error as NSError {
79          resposta = "Falha ao carregar : \(error.localizedDescription)"
80      }
81      return resposta
82  }

```

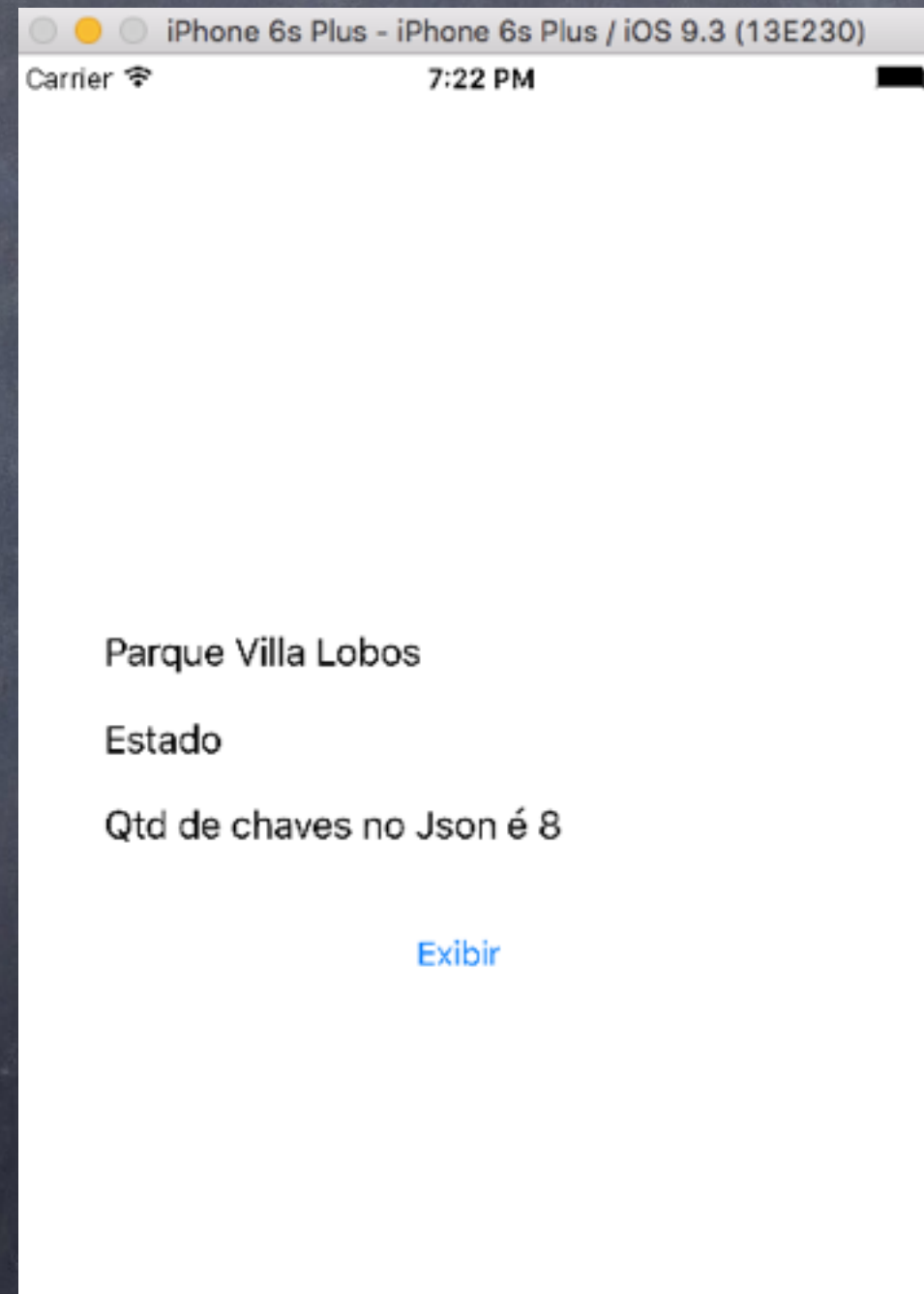

Implementando o Exibir

- O próximo passo é chamar a função dentro da task que roda em segundo plano, comente as linhas 38 e 39 que servia para exibir o conteúdo do Json e implemente as linhas 41 até a 46. O método `dispatch_async` faz com que os dados sejam atualizados na thread principal.

```
33     let task = session!.dataTaskWithURL(url!,
34                                     completionHandler: {
35                                         (data: NSData?, response: NSURLResponse?, error: NSError?) -> Void in
36
37                                         //ações que serão executadas qdo a execução da task é completada
38                                         //let string = NSString(data: data!, encoding: NSUTF8StringEncoding)
39                                         //print(string)
40
41                                         if let nPq = self.retornarNomePq(data!){
42                                             dispatch_async(dispatch_get_main_queue(), {
43                                                 self.local.text = nPq
44                                                 self.qtdInfo.text = "Qtd de chaves no Json é \(self.qtd)"
45                                             })
46                                         }
47                                     })
48     //disparo da execução da task
49     task.resume()
50 }
```


Implementando o Exibir

- Command + R e clique em Exibir, verifique como ficou o resultado ao fazermos rest pelo id, observe que já atualizou o nome do parque e a quantidade de chaves no Json.



Atualizar o estado

- Crie uma função que receba o NSData para que seja feito o parser, e por meio do dicionário com chave "estado" seja possível retornar o nome do estado.

```
77 func retornarEstadoPq(data: NSData) -> String? {  
78     var resposta:String?=nil  
79     do {  
80         //faz leitura dos valore do JSON, NSJSONSerialization faz o Parser do Json  
81         let json = try NSJSONSerialization.JSONObjectWithData(data, options: []) as! [String: AnyObject]  
82         //cria uma string a partir da chave "estado"  
83         if let estadoParque = json["estado"] as? String {  
84             resposta = estadoParque  
85         }  
86     } catch let error as NSError {  
87         resposta = "Falha ao carregar : \(error.localizedDescription)"  
88     }  
89     return resposta  
90 }
```


Atualizar o estado

- Inclua na Task a chamada da função que retorne o estado, existem outras formas para trazer estado fazendo algo semelhante a qtd, por exemplo, incluindo o acesso ao dicionário do estado na mesma função que busca o nome do parque e passando para uma variável o nome do estado, ou até modificando a função para receber o NSData e a chave do dicionário, explore as possibilidades.

```
31 let url = NSURL (string:"https://parks-api.herokuapp.com/parks/577024e4a44821110001ee93")
32 //fazendo o request da url em uma task
33 let task = session!.dataTaskWithURL(url!,
34                                     completionHandler: {
35                                         (data: NSData?, response:NSURLResponse?, error: NSError?) -> Void in
36
37                                         //ações que serão executadas qdo a execução da task é completada
38                                         //let string = NSString(data: data!, encoding: NSUTF8StringEncoding)
39                                         //print(string)
40
41                                         if let nPq = self.retornarNomePq(data!){
42                                             dispatch_async(dispatch_get_main_queue(), {
43                                                 self.local.text = nPq
44                                                 self.qtdInfo.text = "Qtd de chaves no Json é \(self.qtd)"
45                                             })
46                                         }
47
48                                         if let ePq = self.retornarEstadoPq(data!){
49                                             dispatch_async(dispatch_get_main_queue(), {
50                                                 self.estado.text = ePq
51                                             })
52                                         }
53                                     })
54
55 //disparo da execução da task
56 task.resume()
57
58 }
```


Buscar a Foto

- Crie uma função que receba o NSData para que seja feito o parser, e por meio do dicionário com chave "urlfoto" seja possível retornar uma string com a url da foto. Repare que essa função é semelhante as outras.

```
98 func retornarImagemPq(data: NSData) -> String? {
99     var resposta:String?=nil
100     do {
101         //faz leitura dos valore do JSON, NSJSONSerialization faz o Parser do Json
102         let json = try NSJSONSerialization.JSONObjectWithData(data, options: []) as! [String: AnyObject]
103
104         //cria um string a partir da chave "urlfoto"
105         if let urlString = json["urlfoto"] as? String{
106             resposta = urlString
107         }
108     } catch let error as NSError {
109         resposta = "Falha ao carregar : \(error.localizedDescription)"
110     }
111     return resposta
112 }
```

- Crie uma outra função logo abaixo que receba a string com a url da foto e retorne a imagem.

```
114 func carregarImagemdeURL(imageURL : String) {
115     // Cria uma Url de uma string com a URL
116     let url = NSURL(string: imageURL)!
117     // Cria uma taks do tipo Download
118     // - sharedSession = global NSURLCache, NSHTTPCookieStorage and NSURLCredentialStorage objects.
119     let task = NSURLSession.sharedSession().dataTaskWithURL(url) { (responseData, responseUrl, error) -> Void in
120         // se responseData is not null...
121         // recebe o binário da imagem
122         if let imageData = responseData{
123             //transforma o binario em UIImage e atualiza a tela na thread principal
124             dispatch_async(dispatch_get_main_queue(), { () -> Void in
125                 self.Imagem.image = UIImage(data: imageData)
126             })
127         }
128     }
129     //disparo da execução da task
130     task.resume()
131 }
```


Buscar a Foto

- Inclua na Task a chamada da função que retorne a foto.

```
31 let url = NSURL (string:"https://parks-api.herokuapp.com/parks/577024e4a44821110001ee93")
32 //fazendo o request da url em uma task
33 let task = session!.dataTaskWithURL(url!,
34                                     completionHandler: {
35                                         (data: NSData?, response:NSURLResponse?, error: NSError?) -> Void in
36
37                                         //ações que serão executadas qdo a execução da task é completada
38                                         //let string = NSString(data: data!, encoding: NSUTF8StringEncoding)
39                                         //print(string)
40
41                                         if let nPq = self.retornarNomePq(data!){
42                                             dispatch_async(dispatch_get_main_queue(), {
43                                                 self.local.text = nPq
44                                                 self.qtdInfo.text = "Qtd de chaves no Json é \(self.qtd)"
45                                             })
46                                         }
47
48                                         if let ePq = self.retornarEstadoPq(data!){
49                                             dispatch_async(dispatch_get_main_queue(), {
50                                                 self.estado.text = ePq
51                                             })
52                                         }
53
54                                         if let appImageURL = self.retornarImagemPq(data!){
55                                             self.carregarImagemdeURL(appImageURL)
56                                         }
57                                     })
58
59 //disparo da execução da task
60 task.resume()
61 }
62 }
```


Info.plist (forma 1)

- Antes de executar configure o ATS para poder abrir imagens em caminho http:, para isso 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 Loads (5), selecione YES (6).

The screenshot shows the Xcode interface with the following components:

- Left Panel (Project Navigator):** Shows the project structure. The file `Info.plist` is selected and highlighted with a red arrow and a yellow circle labeled **1**.
- Center Panel (Property List Editor):** Displays the `InfoDictionary version` and a list of keys. The key `Supported interface orientations` is selected, and a red arrow points to the '+' button to add a new item, labeled with a yellow circle **2**.
- Right Panel (Property List Editor):** Shows the `App Transport Security Settings` dictionary. A red arrow points to the `App Transport Security Settings` entry, labeled with a yellow circle **3**. Below it, the `App Transport Security Settings` dictionary is expanded, showing a list of options. A red arrow points to the expand/collapse triangle next to the dictionary name, labeled with a yellow circle **4**.
- Bottom Panel (Property List Editor):** Shows the `App Transport Security Settings` dictionary with one item, `Allow Arbitrary Loads`. A red arrow points to the `Allow Arbitrary Loads` entry, labeled with a yellow circle **5**. Another red arrow points to the `YES` value, labeled with a yellow circle **6**.

Resultado

- Execute seu programa, e observe as telas, a segunda tela é do primeiro webservice em php a terceira tela é do segundo webservice em Node.JS.

