

Rest e Json no iOS

X-Code com Swift
Prof. Agesandro Scarpioni
agesandro@fiap.com.br

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

- Objetivo: Criar dois aplicativos para que seja feito um request, depois um parser do Json em um dicionário, para posterior exibição dos 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 (Protocolo Simples de Acesso a Objetos).

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/>

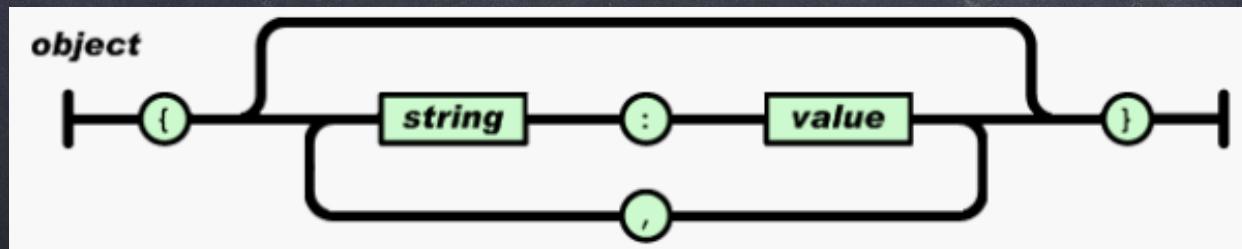
O que é JSON

- JSON é um 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>

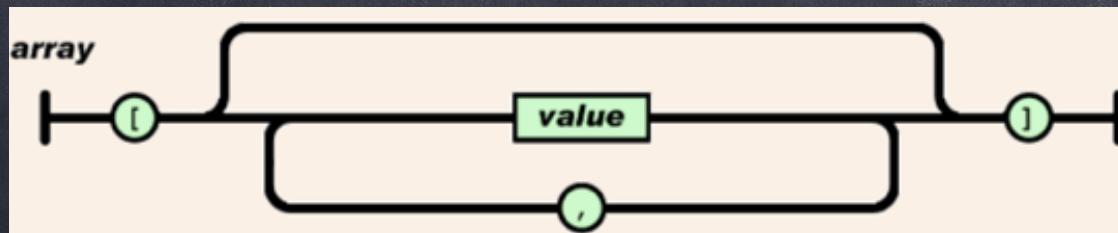
O que é JSON

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

Array**Array**

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 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>

Ajuda para ler o Json

- Json Lint em: <https://jsonlint.com>

JSON JSONLint - The JSON Validator Try the New Pro More Developer Tools

St Adobe Stock
Limited time offer: Get 10 free Adobe Stock images.
ADS VIA CARBON

```
1 {  
2   "abilities": [  
3     "ability": {  
4       "name": "lightning-rod",  
5       "url": "https://pokeapi.co/api/v2/ability/31/"  
6     },  
7     "is_hidden": true,  
8     "slot": 3  
9   },  
10  {  
11    "ability": {  
12      "name": "static",  
13      "url": "https://pokeapi.co/api/v2/ability/9/"  
14    },  
15    "is_hidden": false,  
16    "slot": 1  
17  }],  
18  "base_experience": 112,  
19  "height": 10,  
20  "id": 31,  
21  "is_mythic": false,  
22  "name": "lightning-rod",  
23  "order": 31,  
24  "species": {  
25    "name": "pokemob",  
26    "url": "https://pokeapi.co/api/v2/species/100/"  
27  },  
28  "weight": 10}
```

Validate JSON Clear Support JSONLint for \$2/Month

Ajuda para ler o Json

- Json Viewer em: <http://jsonviewer.stack.hu>

The screenshot shows a JSON viewer interface with two tabs at the top: 'Viewer' (selected) and 'Text'. The main area displays a hierarchical tree of a JSON object. The root node is 'JSON', which contains the following fields:

- abilities
 - base_experience : 112
- forms
- game_indices
 - height : 4
- held_items
 - id : 25
 - is_default : true
 - location_area_encounters : "https://pokeapi.co/api/v2/pokemon/25/encounters"
- moves
 - name : "pikachu"
 - order : 35
- species
- sprites
- stats
- types
- weight : 60

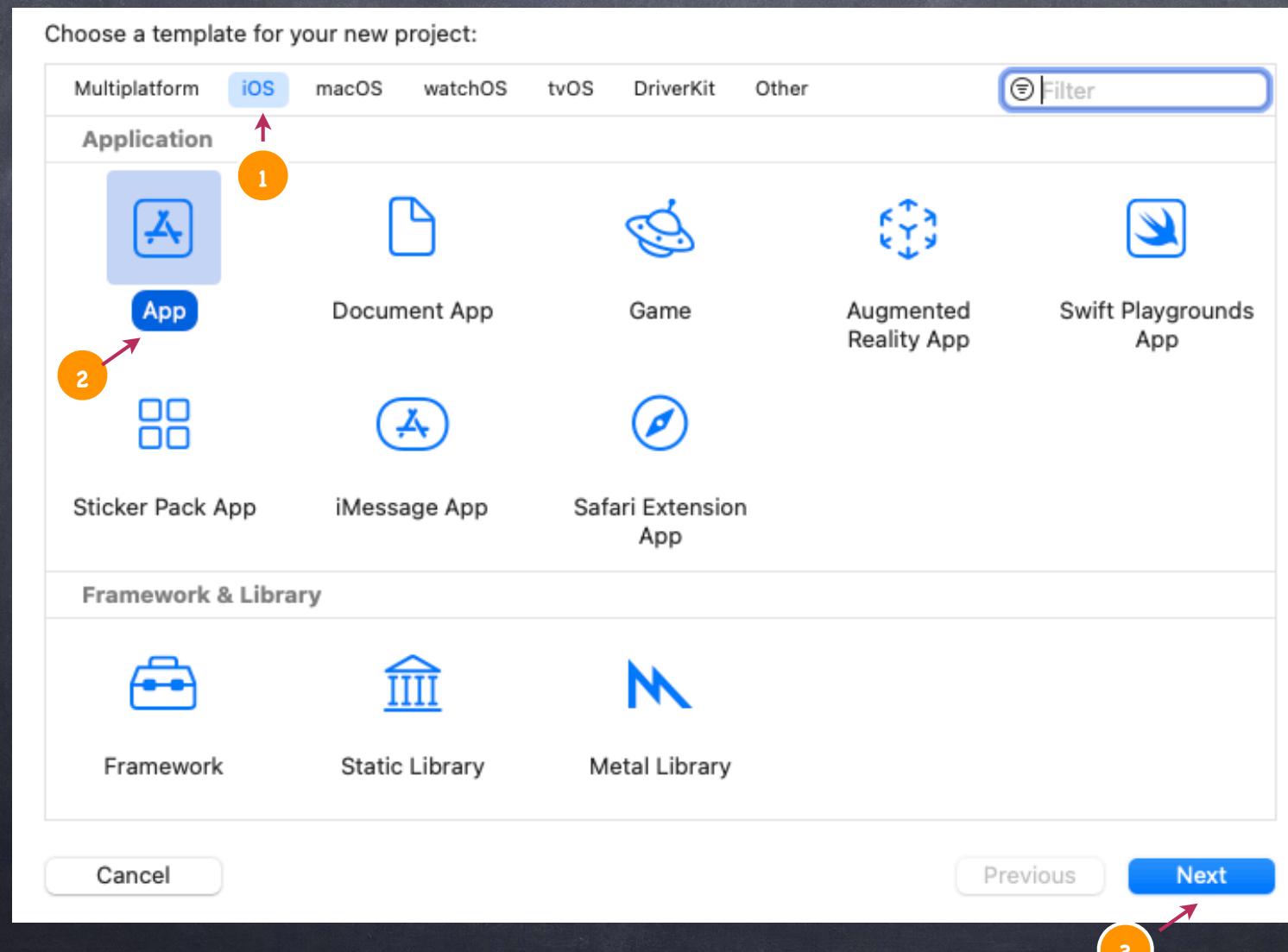
Existem duas formas de fazer o
acesso aos dados:

Raiz ou Nutella, qual você
prefere?

- Para fazer a forma raiz siga da página 11 até 27, se preferir seguir a forma Nutella usando Codable e Decodable disponível desde 2017 no Swift 4, siga da página 28 até 47.

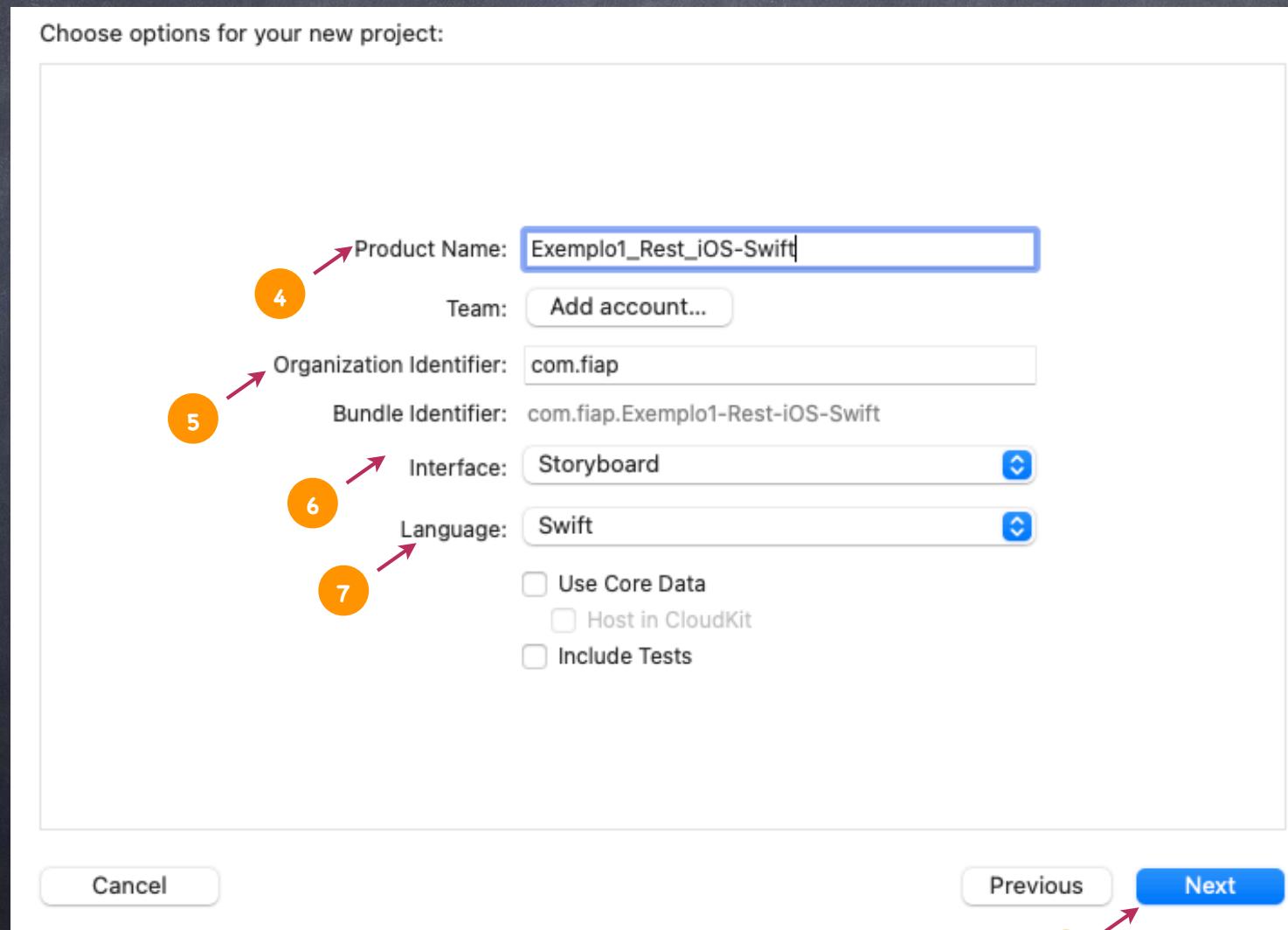
Iniciando o Projeto

- Clique em File -> New Project -> IOS -> App -> Next



Os dados do Projeto

- Preencha com os dados abaixo, em Organization Identifier use uma url invertida, funciona como se fosse o pacote no Java ou o namespace do VB, em Interface escolha Storyboard e em language escolha Swift.



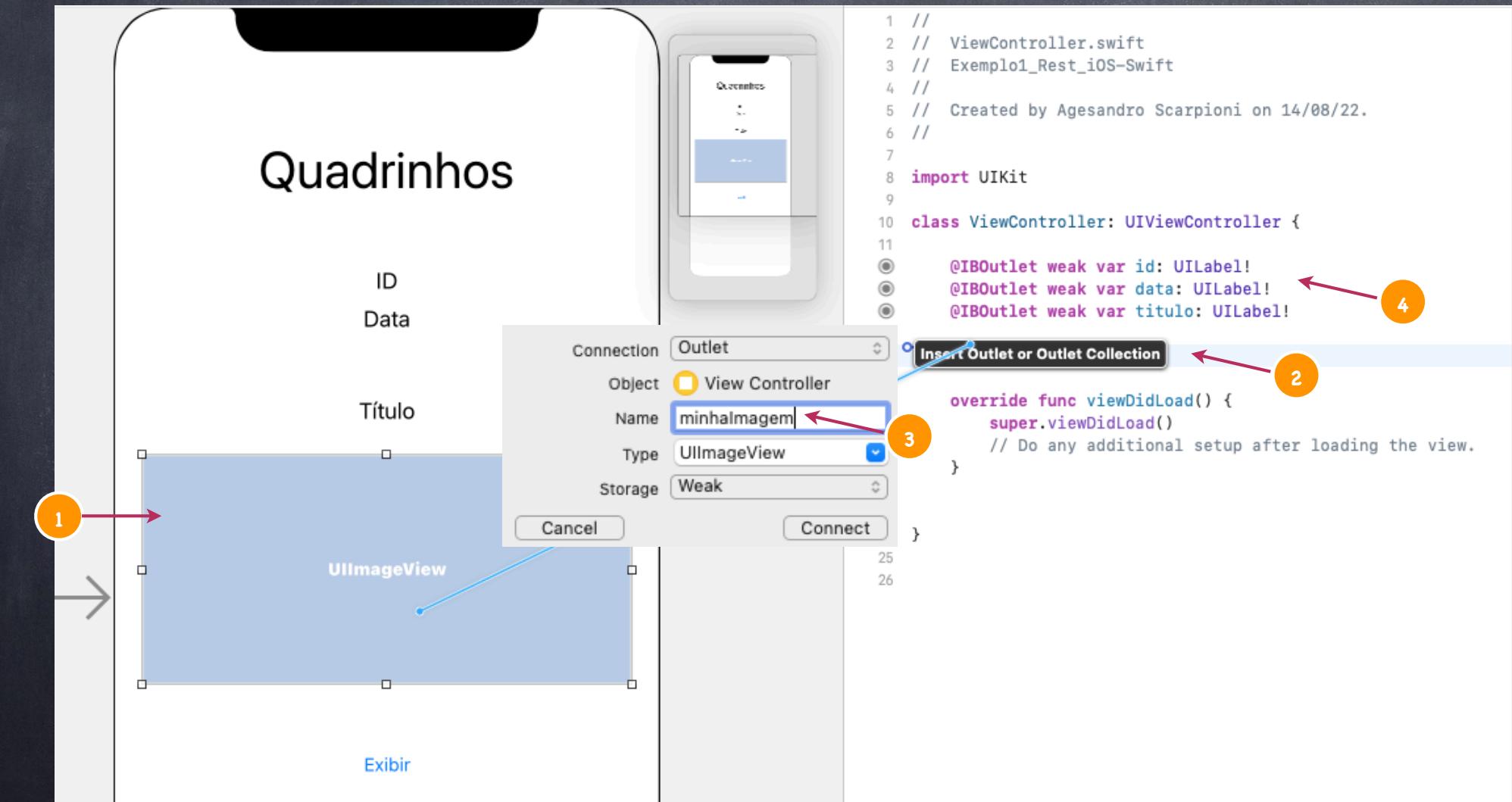
A Interface

- Desenhe a tela abaixo com 4 Label's, 1 ImageView e 1 Button.



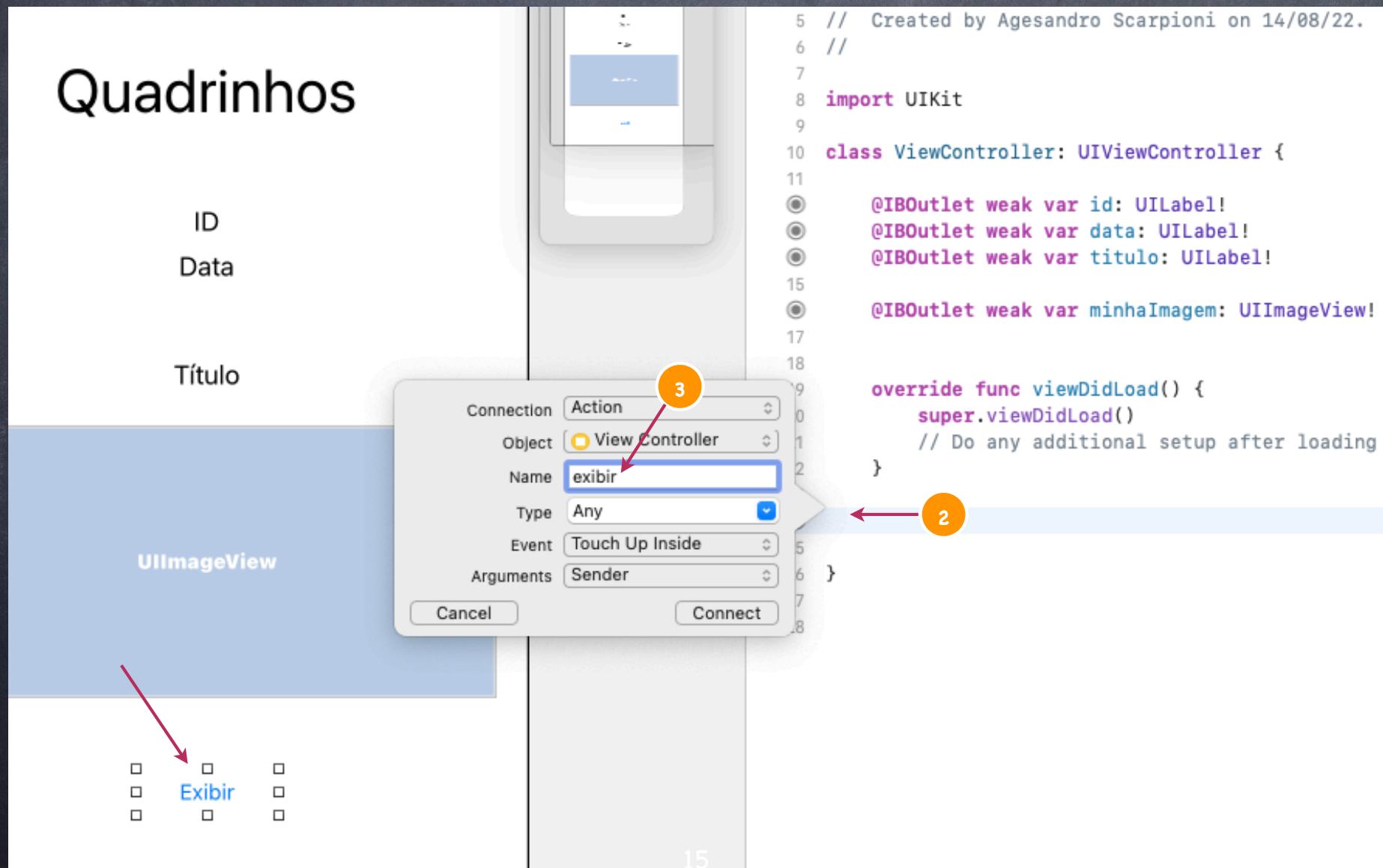
Definindo os IBOOutlet's

No ViewController crie os Outlet's dos Labels (exceto o primeiro) e do ImageView, com botão direito do mouse sobre o objeto e já selecionado (1), arraste até a área indicada (2). Para o ImageView nomeie como minhaImagen(3), para os Label's use respectivamente os nomes id, data e titulo (sem acento) como na imagem (4).



Definindo o IBAction

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



Implementando o Exibir

FIAP

- Nas linhas abaixo será criada uma sessão (linha 18) e a configuração da sessão para realizar o request e posteriormente o parser do Json para um NSDictionary. Para facilitar copie o link do endereço aqui: <http://xkcd.com/info.0.json>

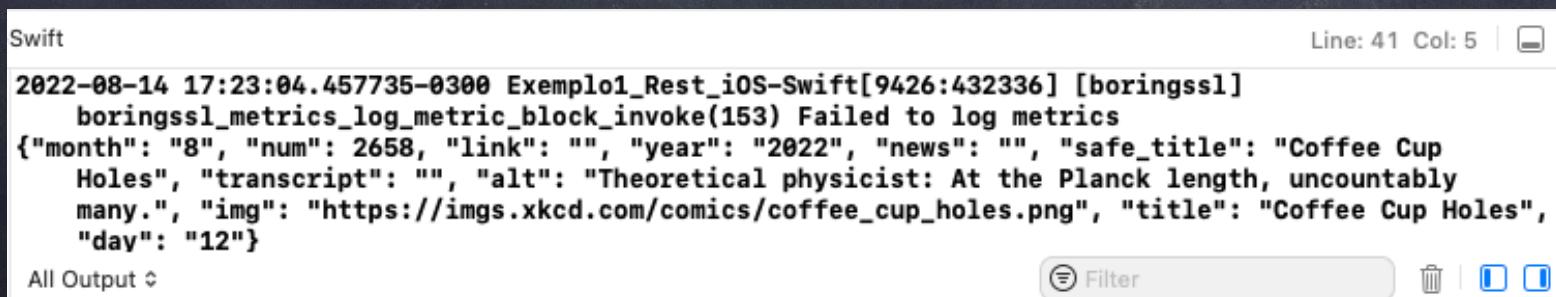
```
1 //  
2 // ViewController.swift  
3 // Exemplo1_Rest_iOS-Swift  
4 //  
5 // Created by Agesandro Scarpioni on 14/08/22.  
6 //  
7  
8 import UIKit  
9  
10 class ViewController: UIViewController {  
11  
12     @IBOutlet weak var id: UILabel!  
12     @IBOutlet weak var data: UILabel!  
12     @IBOutlet weak var titulo: UILabel!  
13  
14     @IBOutlet weak var minhaImagen: UIImageView!  
15  
16     var session: URLSession?  
17  
18     override func viewDidLoad() {  
19         super.viewDidLoad()  
20         // Do any additional setup after loading the view.  
21     }  
22  
23     @IBAction func exibir(_ sender: Any) {  
24         //cria uma configuração de sessão default  
25         let config = URLSessionConfiguration.default  
26         //cria uma sessão com a configuração default  
27         session = URLSession(configuration: config)  
28         //acesso a API  
29         let url = URL(string: "https://xkcd.com/info.0.json") ▲ Initialization of immutable value 'url' was never used;  
30         //  
31     }  
32     |  
33 }  
34 }
```

NSURLSessionTask

- Continuando... -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 33 até a 39

```
25 @IBAction func exibir(_ sender: Any) {
26     //cria uma configuração de sessão default
27     let config = URLSessionConfiguration.default
28     //cria uma sessão com a configuração default
29     session = URLSession(configuration: config)
30     //acesso a API
31     let url = URL(string: "https://xkcd.com/info.0.json")
32
33     let task = session?.dataTask(with: url!, completionHandler: { data, response, error in
34         //ações que serão efetuadas quando a execução do task se completa
35         let texto = NSString(data: data!, encoding: String.Encoding.utf8.rawValue)
36         print(texto!)
37     })
38     //disparo da execução do task
39     task?.resume()
40 }
```

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



The screenshot shows the Xcode interface with the 'Swift' tab selected in the top-left corner. In the bottom-left corner of the main editor area, there is a small 'Output' tab indicator. The main window displays the terminal output. A red arrow points from the text above to the line number 33 in the code block, indicating where the execution starts. The terminal output shows the following JSON data:

```
2022-08-14 17:23:04.457735-0300 Exemplo1_Rest_iOS-Swift[9426:432336] [boringssl]
    boringssl_metrics_log_metric_block_invoke(153) Failed to log metrics
{
    "month": "8", "num": 2658, "link": "", "year": "2022", "news": "", "safe_title": "Coffee Cup
        Holes", "transcript": "", "alt": "Theoretical physicist: At the Planck length, uncountably
        many.", "img": "https://imgs.xkcd.com/comics/coffee_cup_holes.png", "title": "Coffee Cup Holes",
    "day": "12"}
```

Implementando o Exibir

FIAP

- Crie uma função que receba o NSDATA para que seja feito o parser, e por meio do dicionário com chave "title" seja possível retornar o título do quadrinho.

```
42 func retornarTitulo(data:Data)->String?{
43     var resposta:String?=nil
44     do{
45         //A linha abaixo faz a leitura dos valores do Json, NSJSONSerialization faz o parser do Json
46         let json = try JSONSerialization.jsonObject(with: data, options: []) as! [String:Any]
47         //cria e popula uma string a partir da chave neste exemplo "titulo"
48         if let retorno = json["title"] as? String{
49             resposta = retorno
50         }
51     }catch let error as NSError{
52         resposta = "Falha ao carregar \(error.localizedDescription)"
53     }
54     return resposta
55 }
```

Implementando o Exibir

- O próximo passo é chamar a função dentro da task que roda em segundo plano, comente as linhas 35 e 36 que servia para exibir o conteúdo do Json e implemente as linhas 37 até a 41. O método DispatchQueue faz com que os dados sejam atualizados na thread principal.

```
32
33     let task = session?.dataTask(with: url!, completionHandler: { data, response, error in
34         //ações que serão efetuadas quando a execução do task se completa
35         //let texto = NSString(data: data!, encoding: String.Encoding.utf8.rawValue)
36         //print(texto!)
37         if let texto = self.retornarTitulo(data: data!){
38             DispatchQueue.main.sync {
39                 self.titulo.text = texto
40             }
41         }
42     })
43     //disparo da execução do task
44     task?.resume()
45 }
```

Implementando o Exibir

FIAP

- Command + R e clique em Exibir, verifique como ficou o resultado ao fazermos o rest, observe que já atualizou o título do quadrinho.



Exercício

- Existe diversas formas para atualizar os outros campos. Tente atualizar os outros Labels, com o número do quadrinho, o dia, mês e ano.

Buscar a Imagem

FIAP

- Crie uma função que receba o NSDATA para que seja feito o parser, e por meio do dicionário com chave "img" seja possível retornar uma string com a url da imagem.
- Repare que essa função é semelhante a anterior. Pergunta: Daria para fazer algo em relação a isso para deixar o código melhor e com menos repetição?

```
95     func retornarImagenPQ(data:Data) -> String?{
96         var resposta: String? = nil
97         do{
98             //a linha abaixo faz a leitura dos valores, NSJSONSerialization faz o parser do Json
99             let json = try JSONSerialization.jsonObject(with: data, options: []) as! [String: Any]
100            if let urlString = json["urlfoto"] as? String{
101                resposta = urlString
102            }
103        }catch let error as NSError{
104            resposta = "Falha ao carregar \(error.localizedDescription)"
105        }
106        return resposta
107    }
```

Buscar a Imagem

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

```
76 func carregarImagenURL(imagemURL:String){  
77     //cria uma constante para recer a string com a url da imagem, e transforma-la em URL  
78     let myUrl = URL(string: imagemURL)  
79     //cria uma constante para recebrer uma Request da URL  
80     let url = URLRequest(url: myUrl!)  
81     //cria uma task do tipo Download  
82     let session = URLSession.shared  
83     let task = session.dataTask(with: url) { data, response, error in  
84         //se resposta = not null recebe o binário da imagem  
85         if let imagemData = data{  
86             //transforma o binário em UIImage e atualiza a tela na thread principal  
87             DispatchQueue.main.async {  
88                 self.minhaImagen.image = UIImage(data: imagemData)  
89             }  
90         }  
91     }  
92     //disparo da execução da task  
93     task.resume()  
94 }
```

Buscar a Imagem

FIAP

- Inclua na Task a chamada da função que retorna a imagem.

```
31     let url = URL(string: "https://xkcd.com/info.0.json")
32
33     let task = session?.dataTask(with: url!, completionHandler: { data, response, error in
34         //ações que serão efetuadas quando a execução do task se completa
35         //let texto = NSString(data: data!, encoding: String.Encoding.utf8.rawValue)
36         //print(texto!)
37         if let texto = self.retornarTitulo(data: data!){
38             DispatchQueue.main.sync {
39                 self.titulo.text = texto
40             }
41         }
42         if let appImageURL = self.retornarImagem(data: data!){
43             DispatchQueue.main.sync {
44                 self.carregarImagenURL(imagemURL: appImageURL)
45             }
46         }
47     })
48     //disparo da execução do task
49     task?.resume()
50 }
```

Exibindo a imagem

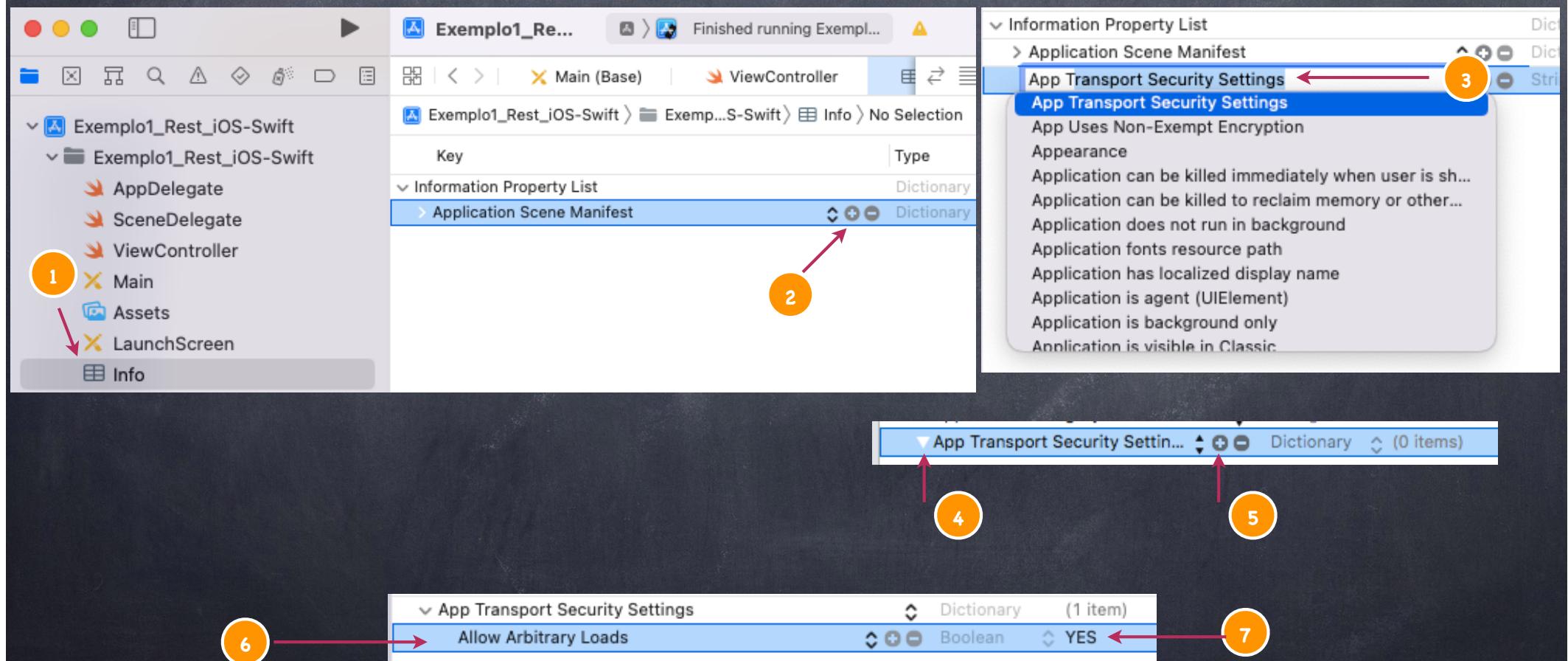
- Ao executar um programa que a url da imagem é http ou invés de https, que não é este caso.
- Observe o erro que ocorre quando você tenta exibir uma imagem utilizando o http ao invés de https, sobre essas informações de segurança veja o slide `WebKitView_Swift` que trata de ATS (App Transport Security).

```
finished with error [-1022] Error Domain=NSURLErrorDomain Code=-1022 "The resource could not be loaded because the App Transport Security policy requires the use of a secure connection." UserInfo={NSLocalizedDescription=The resource could not be loaded because the App Transport Security policy requires the use of a secure connection.,  
NSErrorFailingURLStringKey=http://passeioskids.com/wp-content/uploads/2017/07/IMG_0740.jpg,  
NSErrorFailingURLKey=http://passeioskids.com/wp-content/uploads/2017/07/IMG_0740.jpg,  
_NSURLErrorRelatedURLSessionTaskErrorKey=(  
"LocalDataTask <2B28EED4-4C91-47B7-9A98-3057519A9E42>.<1>"  
, _NSURLErrorFailingURLSessionTaskErrorKey=LocalDataTask <2B28EED4-4C91-47B7-9A98-3057519A9E42>.<1>,  
NSUnderlyingError=0x6000016509c0 {Error Domain=kCFErrorDomainCFNetwork Code=-1022 "(null)"}  
All Output ▾ Filter | ✖ ✖ ✖
```

- Para continuar o exercício veja na próxima página uma das soluções disponíveis no slide da aula `WebKitView_Swift`.

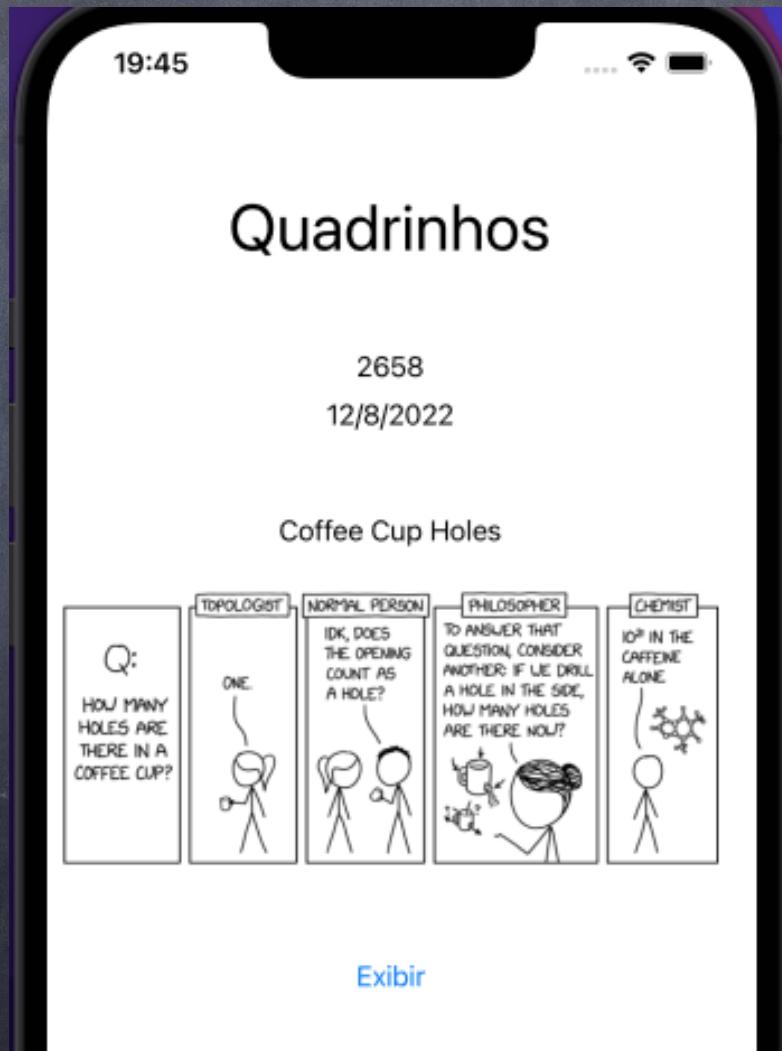
Info.plist

- Antes de executar, configure o ATS para poder abrir imagens caso o caminho de sua imagem seja http:, para isso abra o arquivo info(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 (5), role as opções para encontrar: Allow Arbitrary Loads (6), digite ou troque para YES (7).



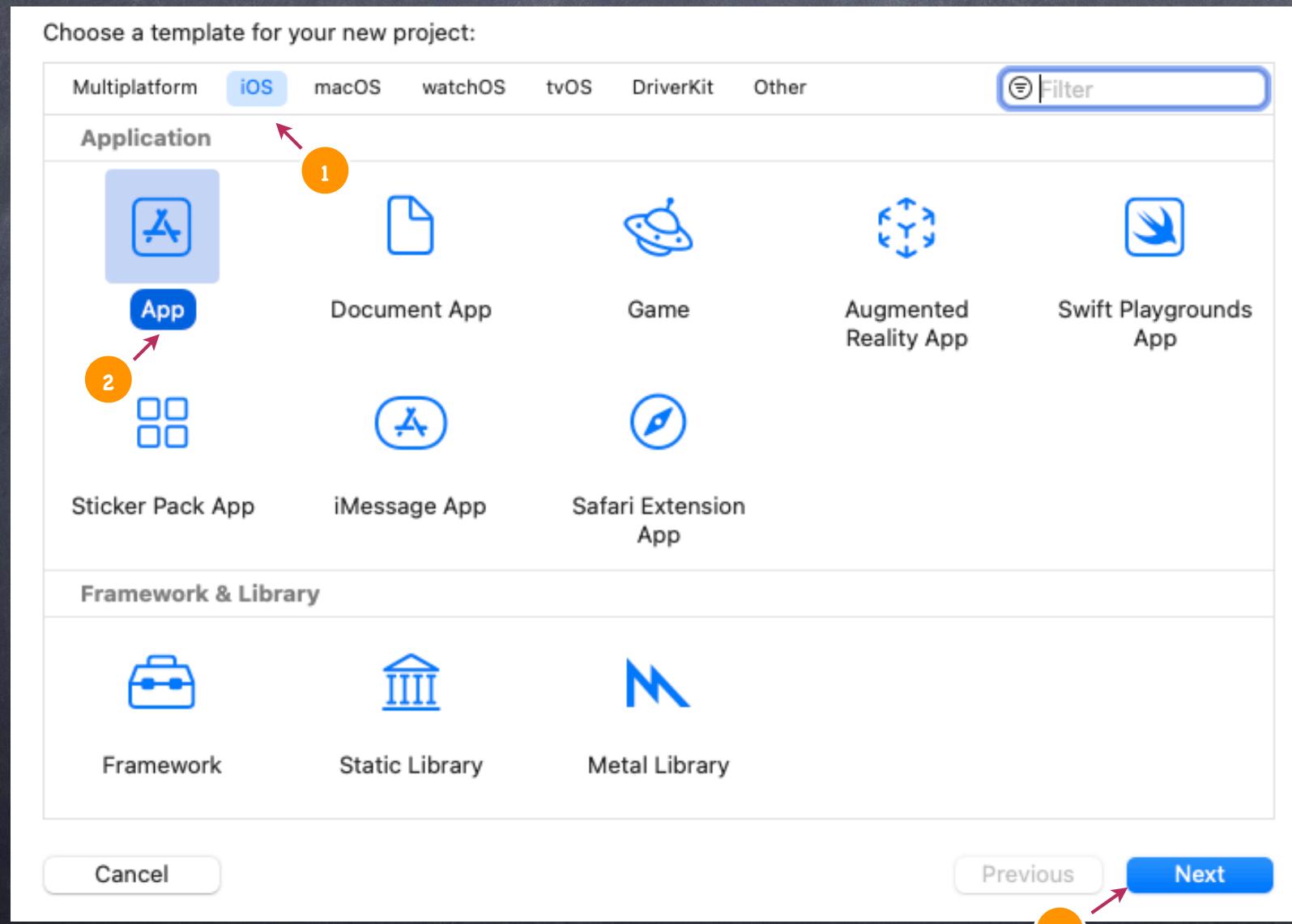
Resultado

- Execute seu programa, e observe a tela.



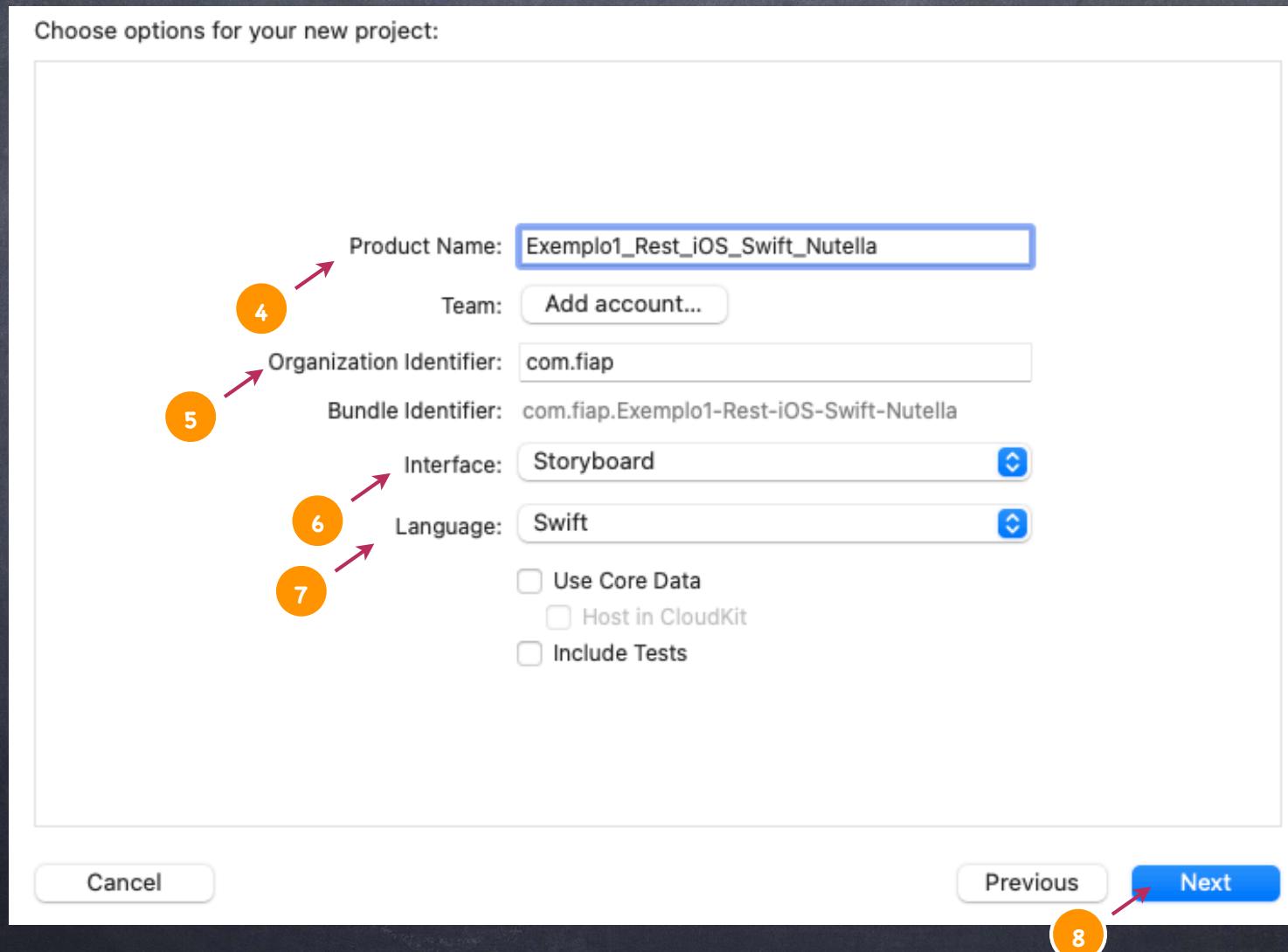
Iniciando o Projeto “Nutela”

- Clique em File -> New Project -> IOS -> App -> Next



Os dados do Projeto

- Preencha com os dados abaixo, lembre-se que o Organization Identifier é uma url invertida e funciona como se fosse o pacote no Java ou o namespace do VB, em Interface escolha Storyboard e em language escolha Swift.



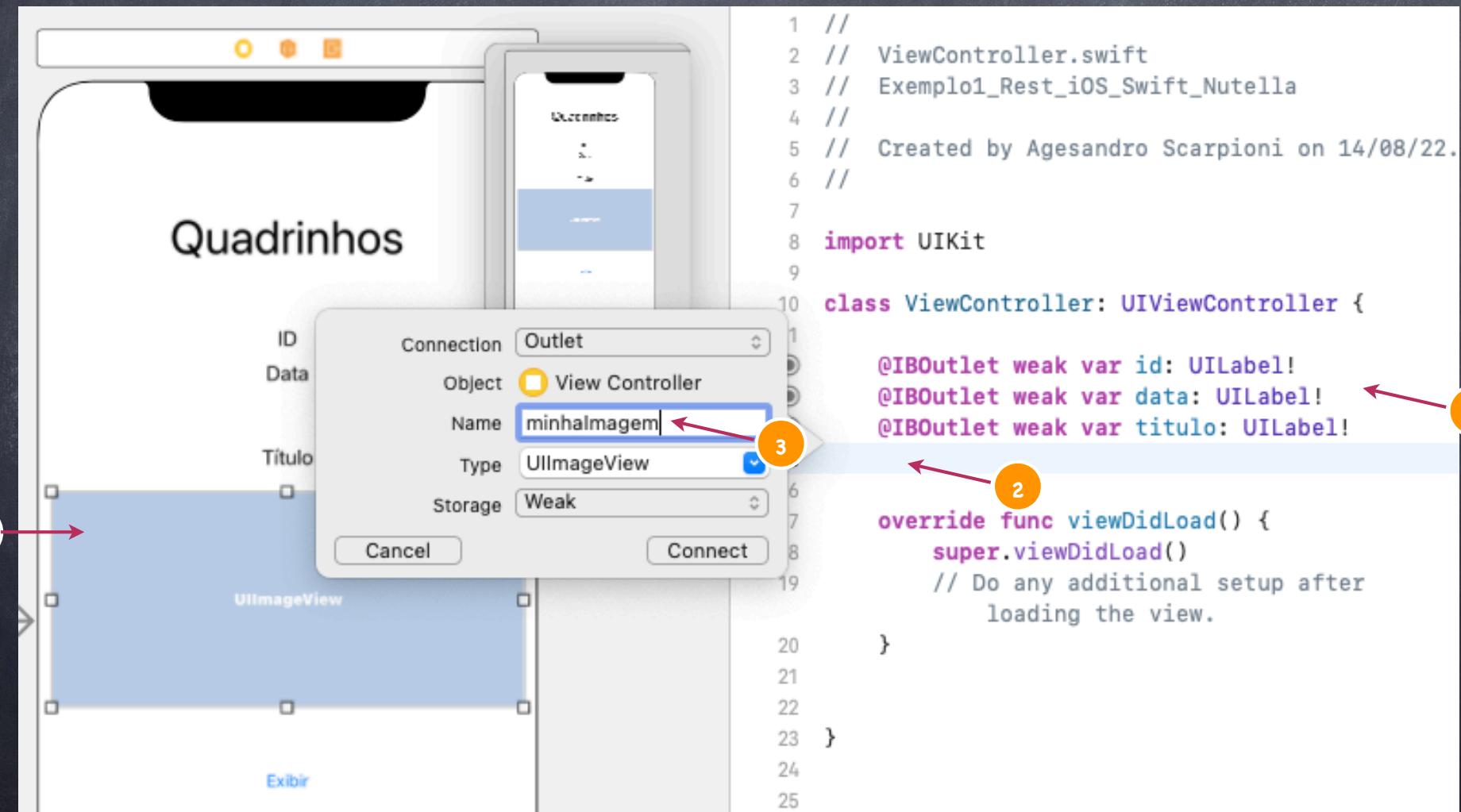
A Interface

- Desenhe a tela abaixo com 4 Label's, 1 ImageView e 1 Button.



Definindo os IBOutlets

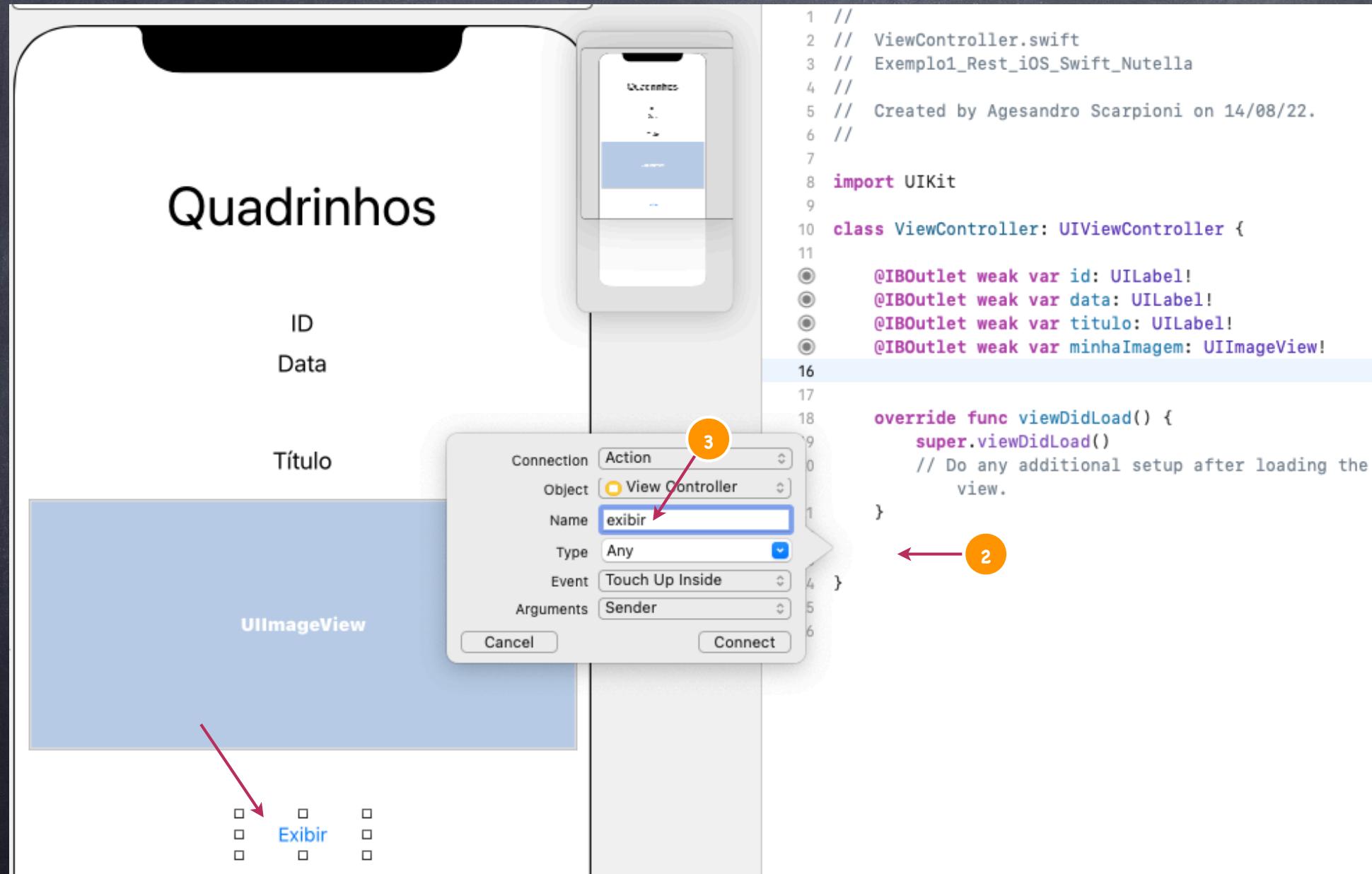
No ViewController crie os Outlet's dos Labels (exceto o primero) e do ImageView, com botão direito do mouse sobre o objeto e já selecionado (1), arraste até a área indicada (2). Para o ImageView nomeie como minhaImagen(3), para os Label's use respectivamente os nomes id, data e titulo (sem acento) como na imagem (4).



Definindo o IBAction

FIAP

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



Outlet's e Action

FIAP

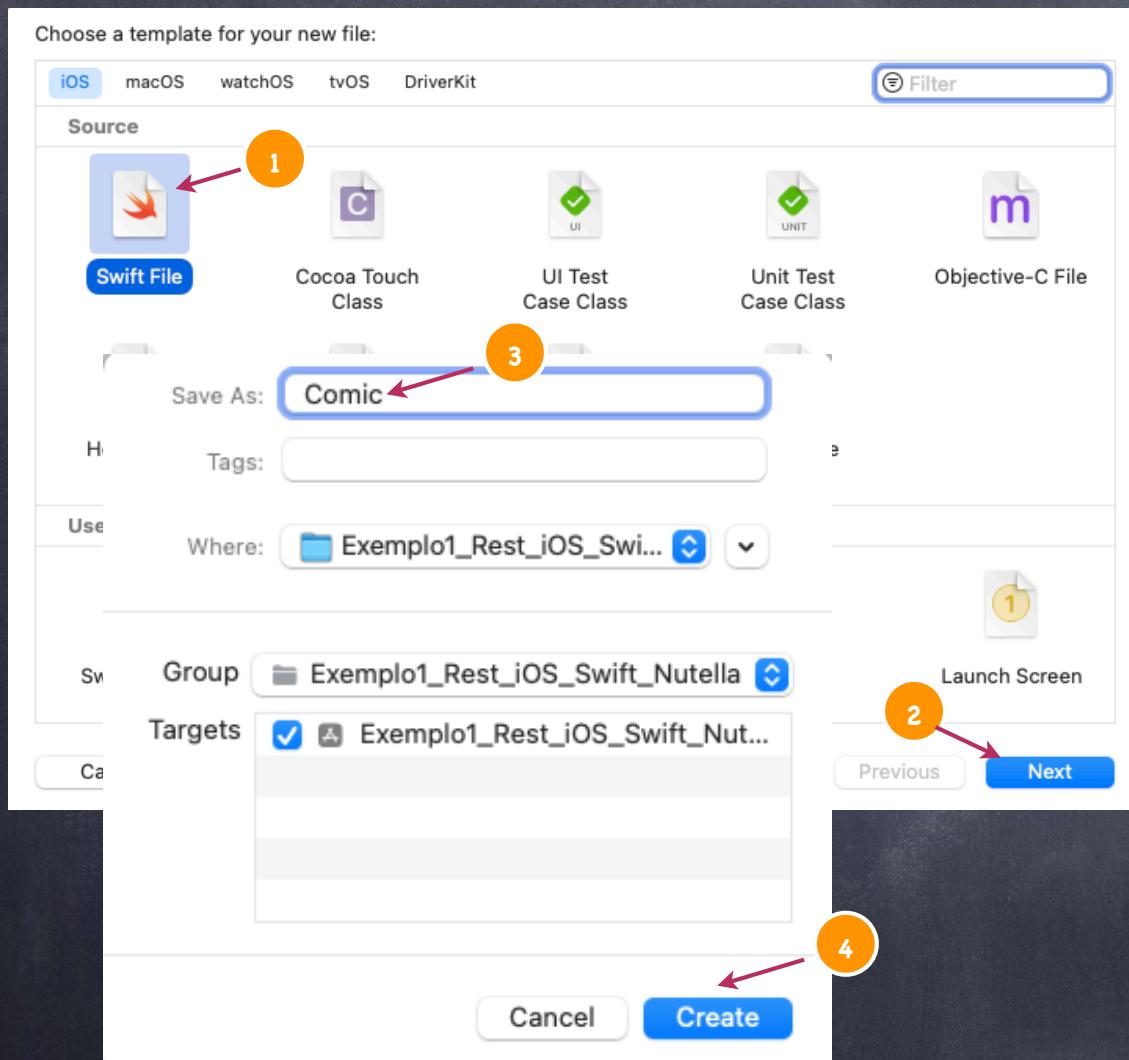
- Veja como ficaram as linhas com todos os Outlets e o Action.

```
1 //  
2 //  ViewController.swift  
3 //  Exemplo1_Rest_iOS_Swift_Nutella  
4 //  
5 //  Created by Agesandro Scarpioni on 14/08/22.  
6 //  
7  
8 import UIKit  
9  
10 class ViewController: UIViewController {  
11  
12     @IBOutlet weak var id: UILabel!  
13     @IBOutlet weak var data: UILabel!  
14     @IBOutlet weak var titulo: UILabel!  
15     @IBOutlet weak var minhaImagem: UIImageView!  
16  
17  
18     override func viewDidLoad() {  
19         super.viewDidLoad()  
20         // Do any additional setup after loading the view.  
21     }  
22  
23     @IBAction func exibir(_ sender: Any) {  
24  
25     }  
26  
27 }
```

Struct

FIAP

- Clique em File -> New -> File e crie um arquivo Swift File(1,2) com o nome Comic(3,4), implemente a estrutura (5) da linha 10 até a 17, esse Struct precisa ter os mesmos nomes e tipos que o Json (6), não é necessário ter todos os campos, mas os campos que você necessitar precisam ter o mesmo nome.



```
1 {  
2     "month": "8",  
3     "num": 2658,  
4     "link": "",  
5     "year": "2022",  
6     "news": "",  
7     "safe_title": "Coffee Cup Holes",  
8     "transcript": "",  
9     "alt": "Theoretical physicist: At the Planck  
10    "img": "https://imgs.xkcd.com/comics/coffee_c  
11    "title": "Coffee Cup Holes",  
12    "day": "12"  
13 }  
  
1 //  
2 // Comic.swift  
3 // Exemplo1_Rest_iOS_Swift_Nutella  
4 //  
5 // Created by Agesandro Scarpioni on 14/08/22.  
6 //  
7  
8 import Foundation  
9  
10 struct Comic: Decodable{  
11     var num:Int  
12     var day:String  
13     var month:String  
14     var year:String  
15     var title:String  
16     var img:String  
17 }  
18 }
```

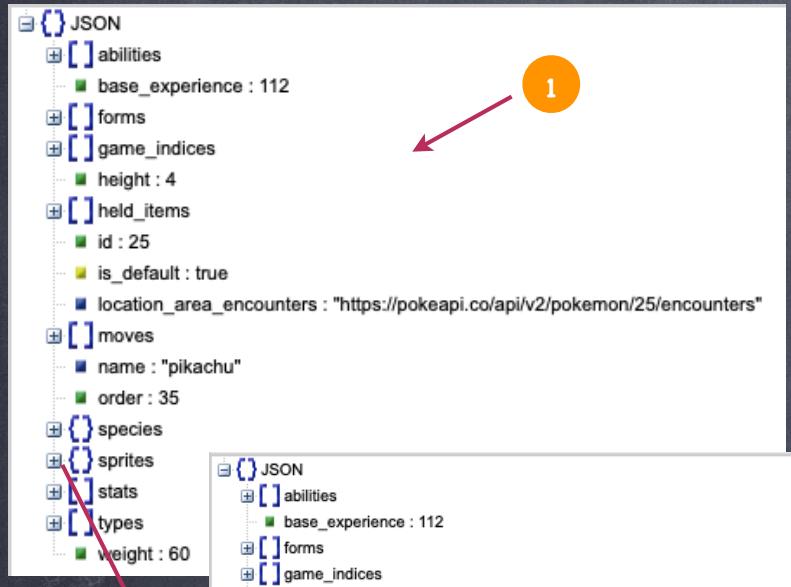
The image shows two parts of a Swift code editor. The top part displays a JSON object with fields like 'month', 'num', 'link', 'year', 'news', etc. The bottom part shows the corresponding Swift struct definition, where the struct name is 'Comic' and it implements the 'Decodable' protocol. The struct contains properties for 'num' (Int), 'day' (String), 'month' (String), 'year' (String), 'title' (String), and 'img' (String). Arrows from numbered circles (1 through 6) point from the UI elements in the Xcode dialog to the corresponding code elements in the editor.

<https://xkcd.com/info.0.json>

Struct para Json complexo

FIAP

Em situações onde é necessário buscar dados dentro de outras estruturas temos que criar um Struct como sub item do struct principal. Veja o exemplo de Json abaixo (1), você irá acessar da mesma forma que o slide anterior os campos base_experience, height, id, is_default, location_area_encounters, name, order, weight (2) que estão na raiz do Json, no entanto, para buscar a imagem do pokemon você precisa entrar na estrutura de sprites(3), para criar a sub estrutura com um nome não real (4) e indicar com o nome real uma chamada à essa estrutura (5), criando um link entre elas(6).



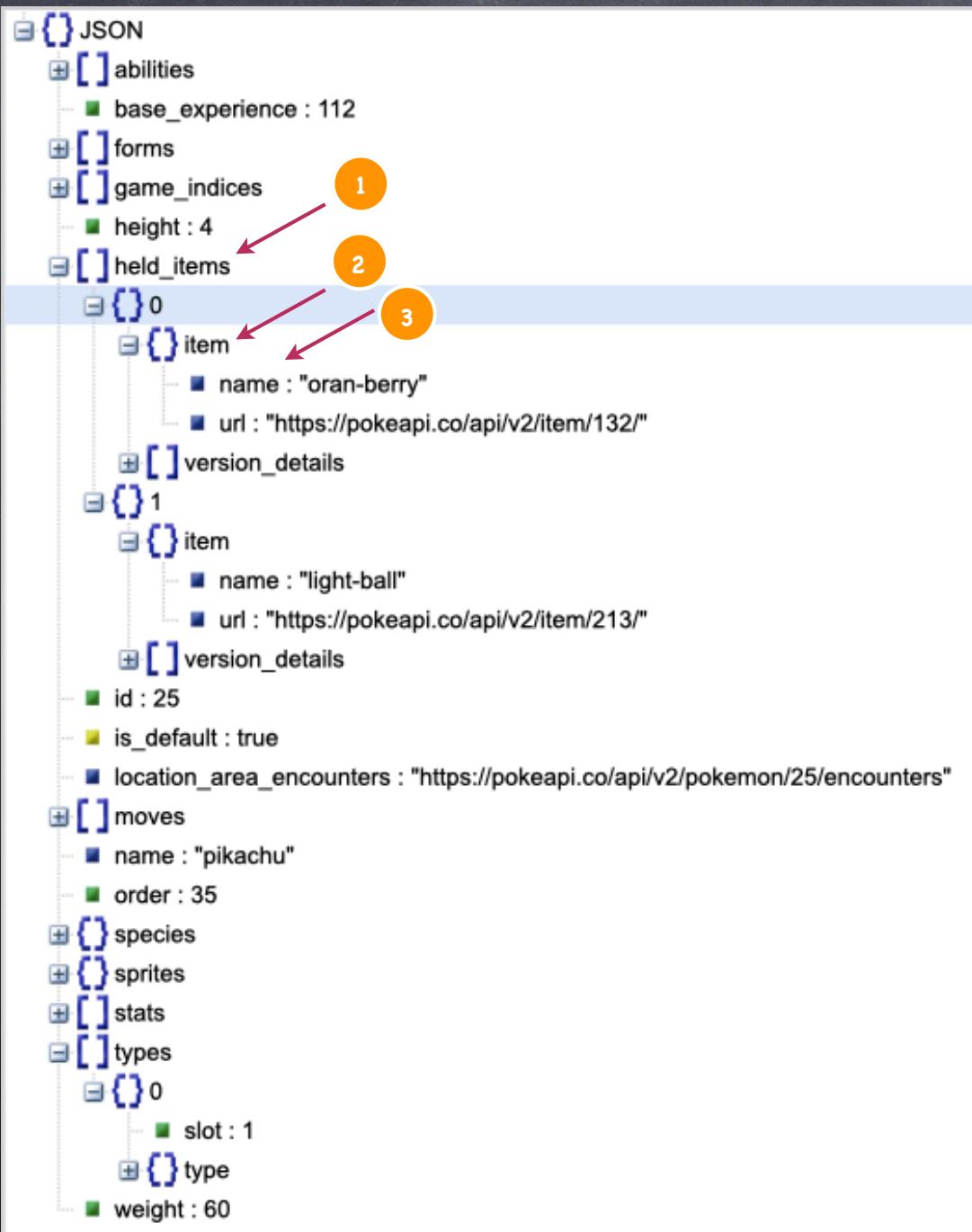
```
9 import Foundation
10
11 struct UmPokemon:Decodable {
12     var id: Int
13     var name:String
14     var order:Int
15     var weight:Int
16     var height:Int
17
18 }
```

```
9 import Foundation
10
11 struct UmPokemon:Decodable {
12     var id: Int
13     var name:String
14     var order:Int
15     var weight:Int
16     var height:Int
17     var sprites:Sprites
18 }
19
20 struct Sprites: Decodable{
21     var front_default:String
22     var front_female:String
23     var back_default:String
24     var back_female:String
25 }
```

Struct para Json complexo

FIAP

Um outro exemplo, porém, neste caso o interesse é buscar o name de item em held_items, veja as relações 1,2 e 3.



```
9 import Foundation
10
11 struct UmPokemon:Decodable {
12     var id: Int
13     var name:String
14     var order:Int
15     var weight:Int
16     var height:Int
17     var sprites:Sprites
18     var held_items:[Held]
19 }
20
21 struct Sprites: Decodable{
22     var front_default:String
23     var front_female:String
24     var back_default:String
25     var back_female:String
26 }
27
28 struct Held:Decodable{
29     var item:Item
30 }
31
32 struct Item:Decodable{
33     var name:String
34 }
```

Struct para Json complexo

FIAP

Exemplo para Exibir os Dados do Array

```
36 func loadJson(){
37     let jsonUrlString = "https://pokeapi.co/api/v2/pokemon/" + campo
38     let url = URL(string: jsonUrlString)
39
40     URLSession.shared.dataTask(with: url!) { (data, response, error) in
41         guard let data = data else {return}
42         do{
43             pokemon = try JSONDecoder().decode(UmPokemon.self, from: data)
44             let foto = self.carregarImagem(enderecoDaFoto: pokemon.sprites.front_default)
45             let itemZero = pokemon.held_items[0].item ← 1
46
47             DispatchQueue.main.sync {
48                 self.labelNome.text = pokemon.name
49                 self.labelNumero.text = String(pokemon.id)
50                 self.minhaImagem.image = foto
51                 self.labelTeste1.text = itemZero.name ← 2
52             }
53         |
54         }catch let jsonError{
55             print("Erro de serialização no Json", jsonError)
56         }
57     }
58     .resume()
59 }
```

Voltando ao projeto "Nutella" - URL

FIAP

- Crie como exibido na linha 10 uma variável Optional comic do tipo Comic(1). Faça a função loadComic() como mostrado na linha 24 (2), e implemente as linhas 25 e 26. Para facilitar copie o link do endereço abaixo:

- <https://xkcd.com/info.0.json>

```
1 //  
2 //  ViewController.swift  
3 //  Exemplo1_Rest_iOS_Swift_Nutella  
4 //  
5 // Created by Agesandro Scarpioni on 14/08/22.  
6 //  
7  
8 import UIKit  
9  
10 var comic:Comic?=nil  
11  
12 class ViewController: UIViewController {  
13  
14     @IBOutlet weak var id: UILabel!  
15     @IBOutlet weak var data: UILabel!  
16     @IBOutlet weak var titulo: UILabel!  
17     @IBOutlet weak var minhaImagem: UIImageView!  
18  
19     override func viewDidLoad() {  
20         super.viewDidLoad()  
21         // Do any additional setup after loading the view.  
22     }  
23  
24     func loadComic(){  
25         let jsonUrlString = "https://xkcd.com/info.0.json"  
26         let url = URL(string: jsonUrlString) ▲ Initialization of im  
27     }  
28  
29     @IBAction func exibir(_ sender: Any) {  
30     }  
31     }  
32  
33 }
```

Task e Do Catch

FIAP

- Crie uma task para acessar a url! e gerar o arquivo data binário com o resultado da url. Linhas 28 a 31.

```
24 func loadComic(){
25     let jsonUrlString = "https://xkcd.com/info.0.json"
26     let url = URL(string: jsonUrlString)
27
28     URLSession.shared.dataTask(with: url!) { data, response, error in
29         code
30     }
31     .resume()
32 }
```

- Na linha 29 faça o teste do conteúdo do data, prepare o "do catch" presente nas linhas 31 até 35.

```
24 func loadComic(){
25     let jsonUrlString = "https://xkcd.com/info.0.json"
26     let url = URL(string: jsonUrlString)
27
28     URLSession.shared.dataTask(with: url!) { data, response, error in
29         guard let data = data else {return} ⚠ Value 'data' was defined b
30
31         do {
32
33             }catch let jsonError{
34                 print("Error serialization Json", jsonError)
35             }
36         }
37         .resume()
38 }
```

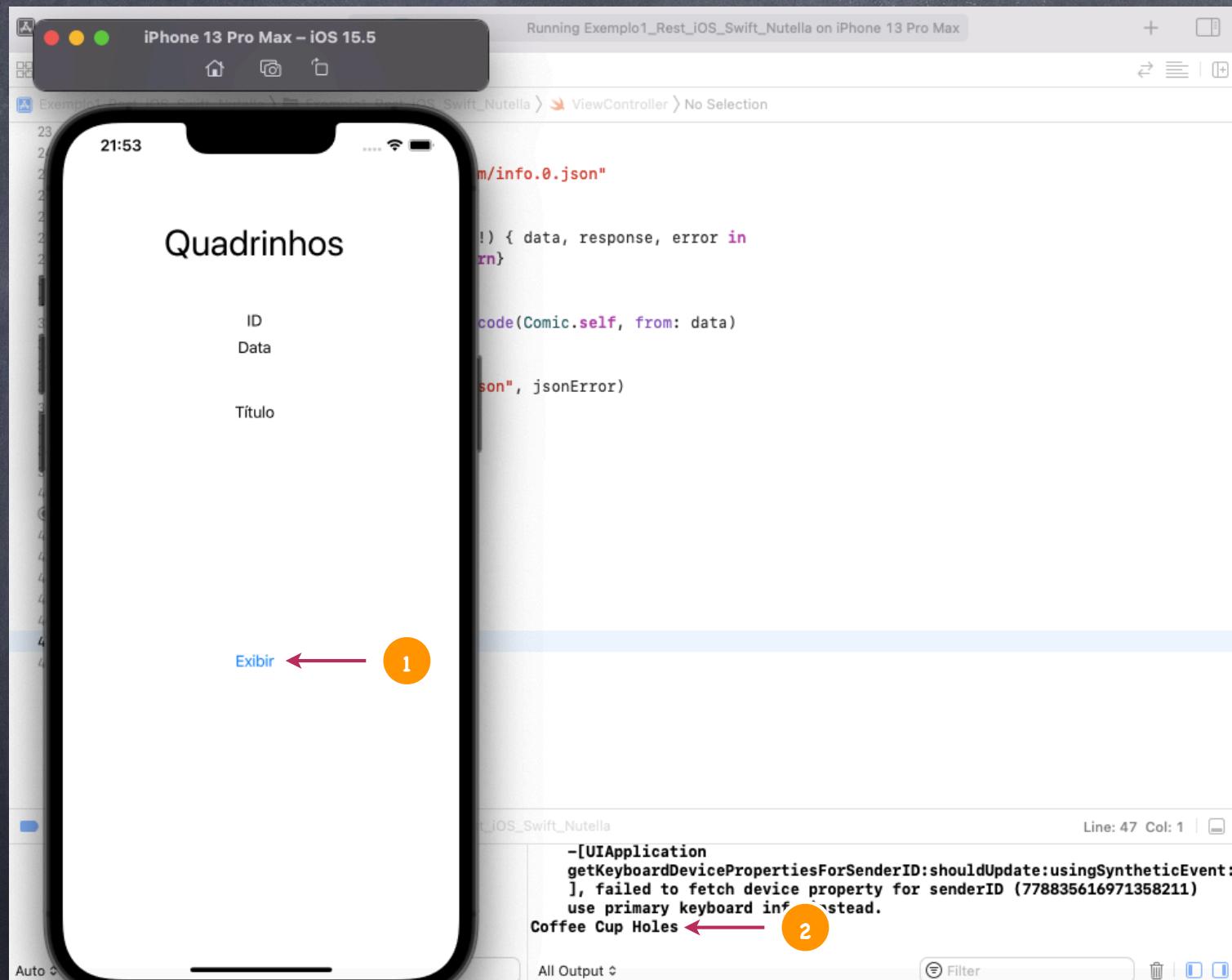
Decode

FIAP

- Depois do "do catch" pronto, programe a linha 32 33 para fazer o decode do data diretamente para a variável "comic", faça o print na linha 33 para testar se já aparece no console o título dos quadrinhos.
- Lembre-se de no Action do exibir, chamar a função loadComic, veja linha 42.

```
24 func loadComic(){
25     let jsonUrlString = "https://xkcd.com/info.0.json"
26     let url = URL(string: jsonUrlString)
27
28     URLSession.shared.dataTask(with: url!) { data, response, error in
29         guard let data = data else {return}
30
31         do {
32             comic = try JSONDecoder().decode(Comic.self, from: data)
33             print(comic.title)
34         }catch let jsonError{
35             print("Error serialization Json", jsonError)
36         }
37     }
38     .resume()
39 }
40
41 @IBAction func exibir(_ sender: Any) {
42     loadComic()
43 }
44
45 }
```

- Quando você executar o projeto e clicar no botão exibir, irá aparecer um título qualquer (2). Agora que o teste funcionou é só fazer a implementação para aparecer os dados nas respectivas labels.



- Comente a linha 33, inclua as linhas 34 a 38, teste seu programa e veja os dados aparecerem na tela.

```
24 func loadComic(){
25     let urlString = "https://xkcd.com/info.0.json"
26     let url = URL(string: urlString)
27
28     URLSession.shared.dataTask(with: url!) { data, response, error in
29         guard let data = data else {return}
30
31         do {
32             comic = try JSONDecoder().decode(Comic.self, from: data)
33             //print(comic.title) ← 1
34             DispatchQueue.main.sync { ← 2
35                 self.titulo.text = comic.title ← 3
36                 self.id.text = String(comic.num) ← 4
37                 self.data.text = comic.day + "/" + comic.month + "/" + comic.year ← 5
38             } ← 6
39
40         }catch let jsonError{
41             print("Error serialization Json", jsonError)
42         }
43     }
44     .resume()
45 }
```

Buscar a Imagem

- Crie a função carregarImagen e que receba a URL da imagem

```
56 func carregarImagen (urlImagen:String) -> UIImage?{  
57     //Obtendo uma url válida  
58     guard let url = URL(string: urlImagen)  
59     else{  
60         print("Não foi possível criar a URL")  
61         return nil  
62     }  
63     var image: UIImage? = nil  
64     do{  
65         //Obtendo os dados válidos  
66         let data = try Data(contentsOf: url)  
67         //Criando a imagem  
68         image = UIImage(data: data)  
69     }catch{  
70         print(error.localizedDescription)  
71     }  
72     return image  
73 }
```

Buscar a Imagem

- Inclua em loadComic a chamada da função (1) e a passagem da foto para o Outlet (2).

```
24 func loadComic(){
25     let urlString = "https://xkcd.com/info.0.json"
26     let url = URL(string: urlString)
27
28     URLSession.shared.dataTask(with: url!) { data, response, error in
29         guard let data = data else {return}
30
31         do {
32             comic = try JSONDecoder().decode(Comic.self, from: data)
33             //print(comic.title)
34             let imagem = self.carregarImagem(urlImagen: comic.img) ← 1
35             DispatchQueue.main.sync {
36                 self.titulo.text = comic.title
37                 self.id.text = String(comic.num)
38                 self.data.text = comic.day + "/" + comic.month + "/" + comic.year
39                 self.minhaImagen.image = imagem ← 2
40             }
41
42         }catch let jsonError{
43             print("Error serialization Json", jsonError)
44         }
45     }
46     .resume()
47 }
```

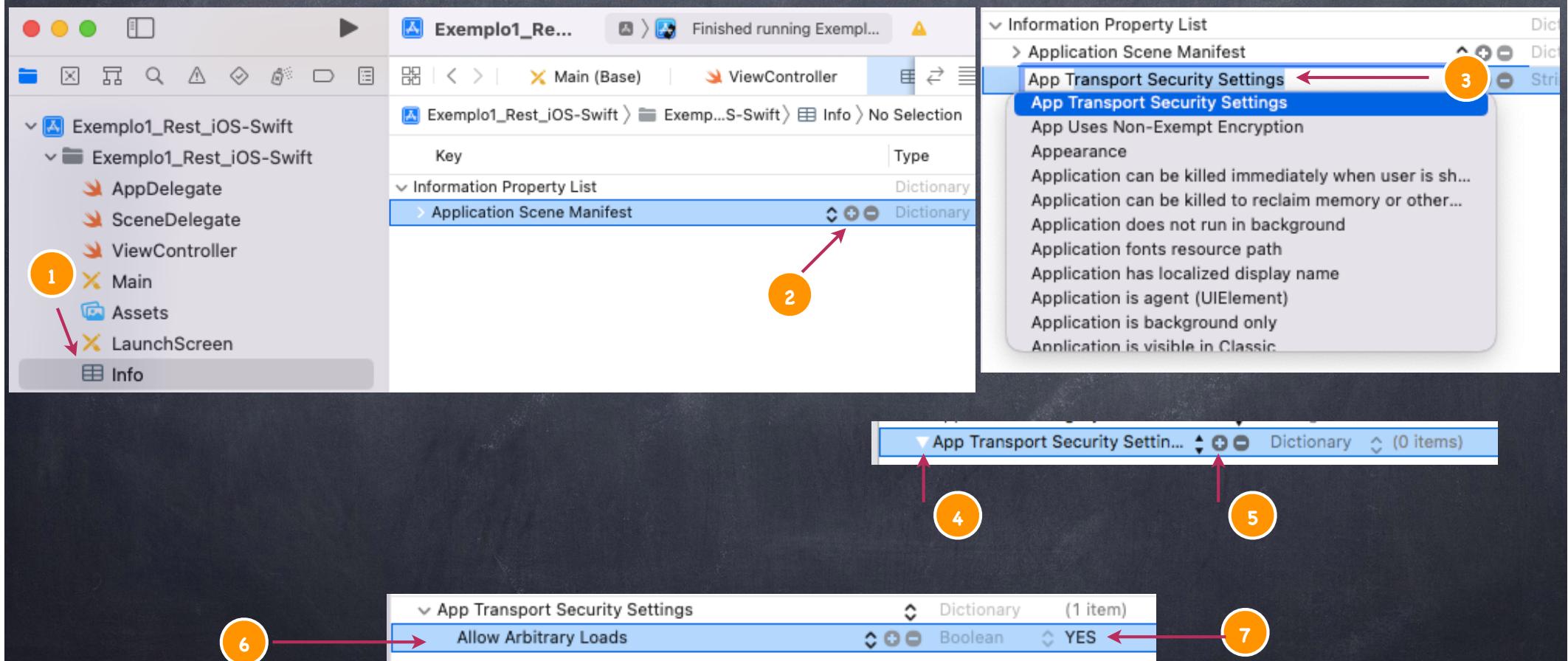
Run

- Caso as imagens estejam em uma URL http ao invés de https, ao executar sua aplicação irá aparecer um erro de ATS (App Transport Security), para ajustar isso, execute os passos do próximo slide. No entanto não é o caso desse exemplo porque as imagens estão em uma url HTTPS.

```
2020-09-25 16:56:37.329182-0300
Exemplo1_Rest_iOS_Swift_Nutella[8093:491280] 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.
2020-09-25 16:56:37.329347-0300
Exemplo1_Rest_iOS_Swift_Nutella[8093:491280] Cannot start
load of Task <42E9C0F2-F319-4291-9105-8C3F0D3D7112>.<0>
since it does not conform to ATS policy
2020-09-25 16:56:37.329512-0300
Exemplo1_Rest_iOS_Swift_Nutella[8093:491285]
NSURLConnection finished with error - code -1022
The file "IMG_0740.jpg" couldn't be opened.
```

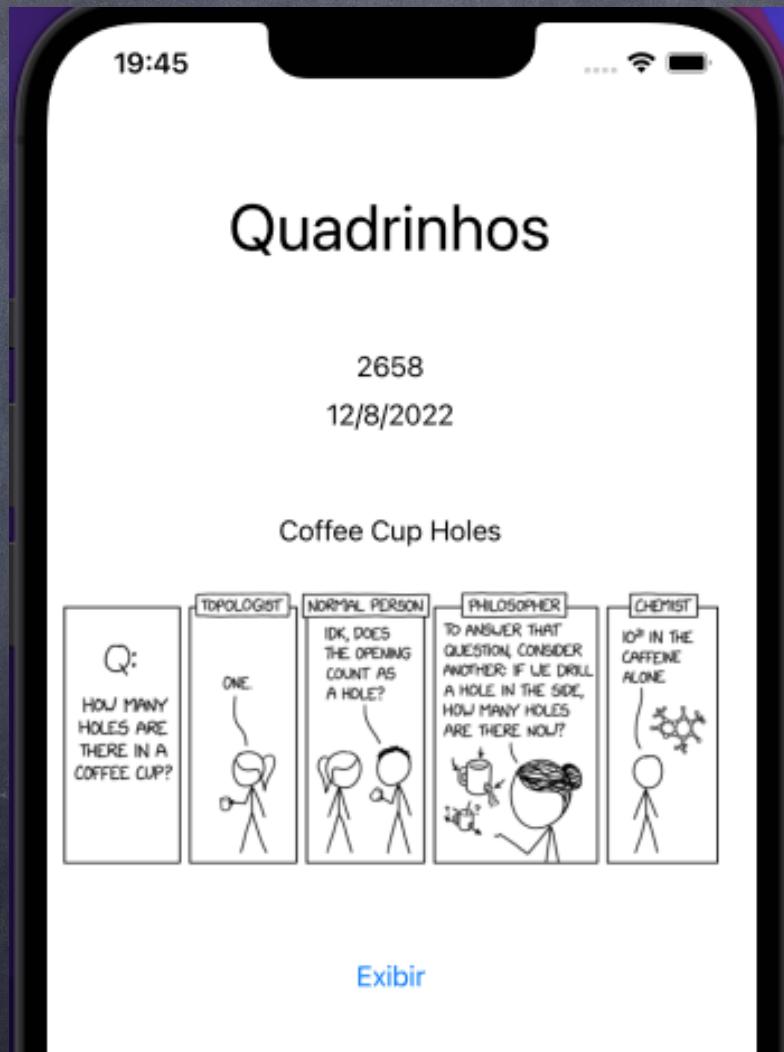
Info.plist

- Antes de executar, configure o ATS para poder abrir imagens caso o caminho de sua imagem seja http:, para isso abra o arquivo info(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 (5), role as opções para encontrar: Allow Arbitrary Loads (6), digite ou troque para YES (7).



Run

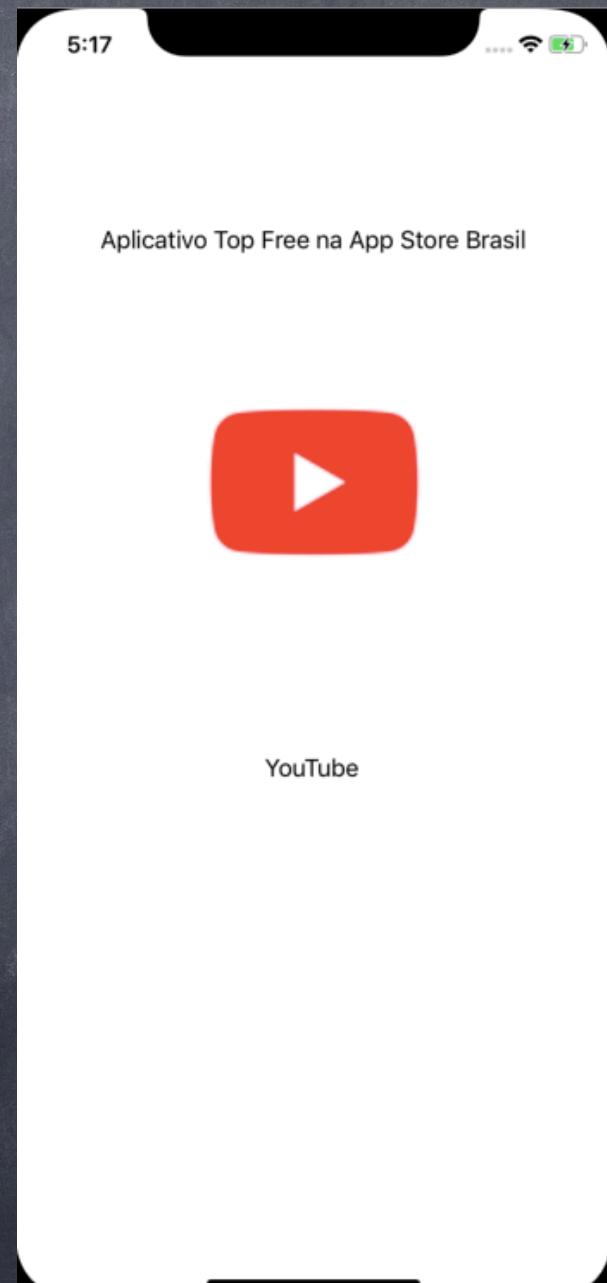
- Execute seu programa, e observe a tela.



Exercício

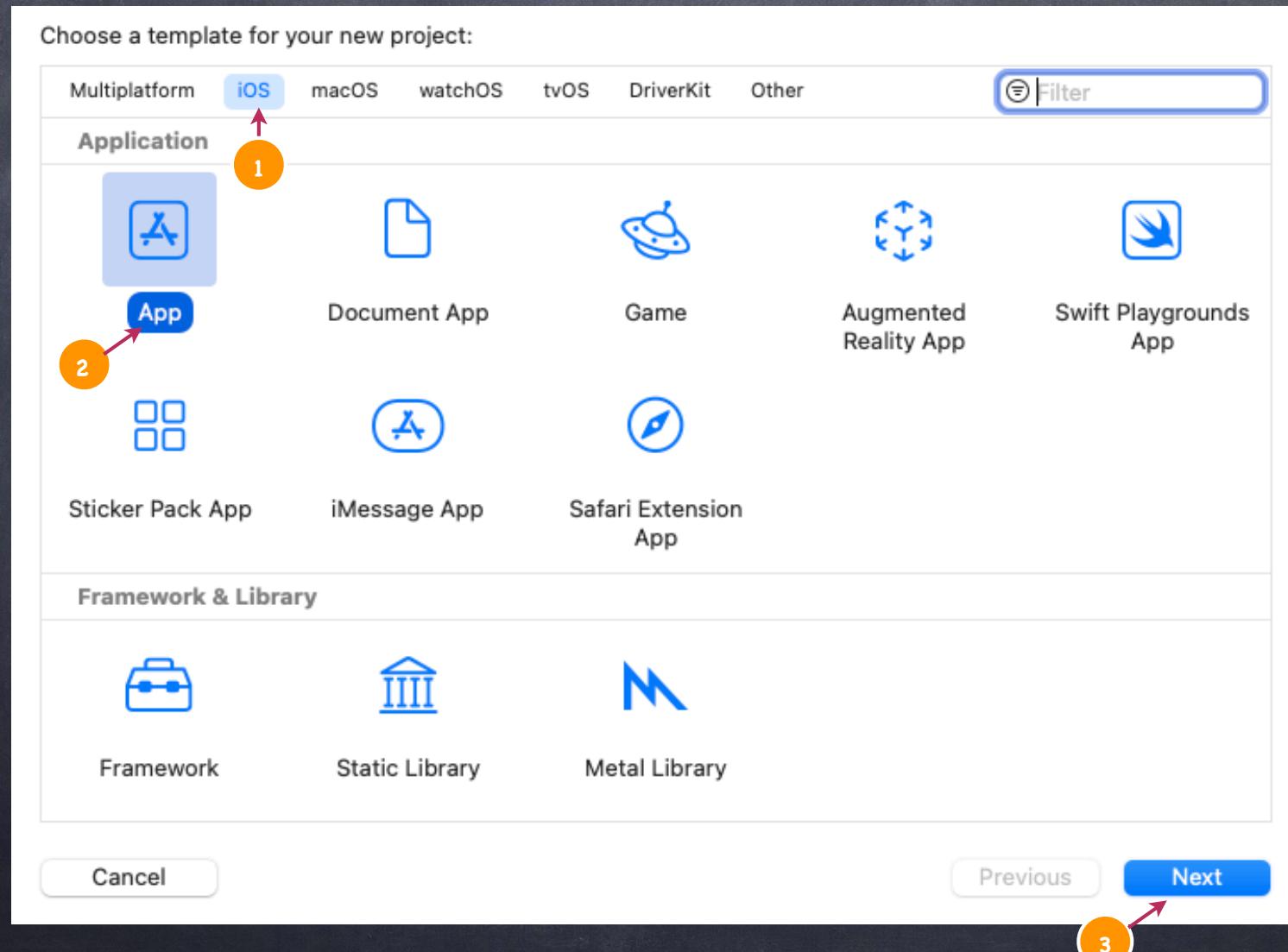
FIAP

- Na próximas páginas você irá desenvolver um projeto mais complexo, todo o slide está guiado para a forma “Raiz”, no entanto, é sua missão com base no conhecimento de Decodable, exibir o top free aplicativo no site da Apple utilizando a forma “Nutella” para ter o resultado abaixo :



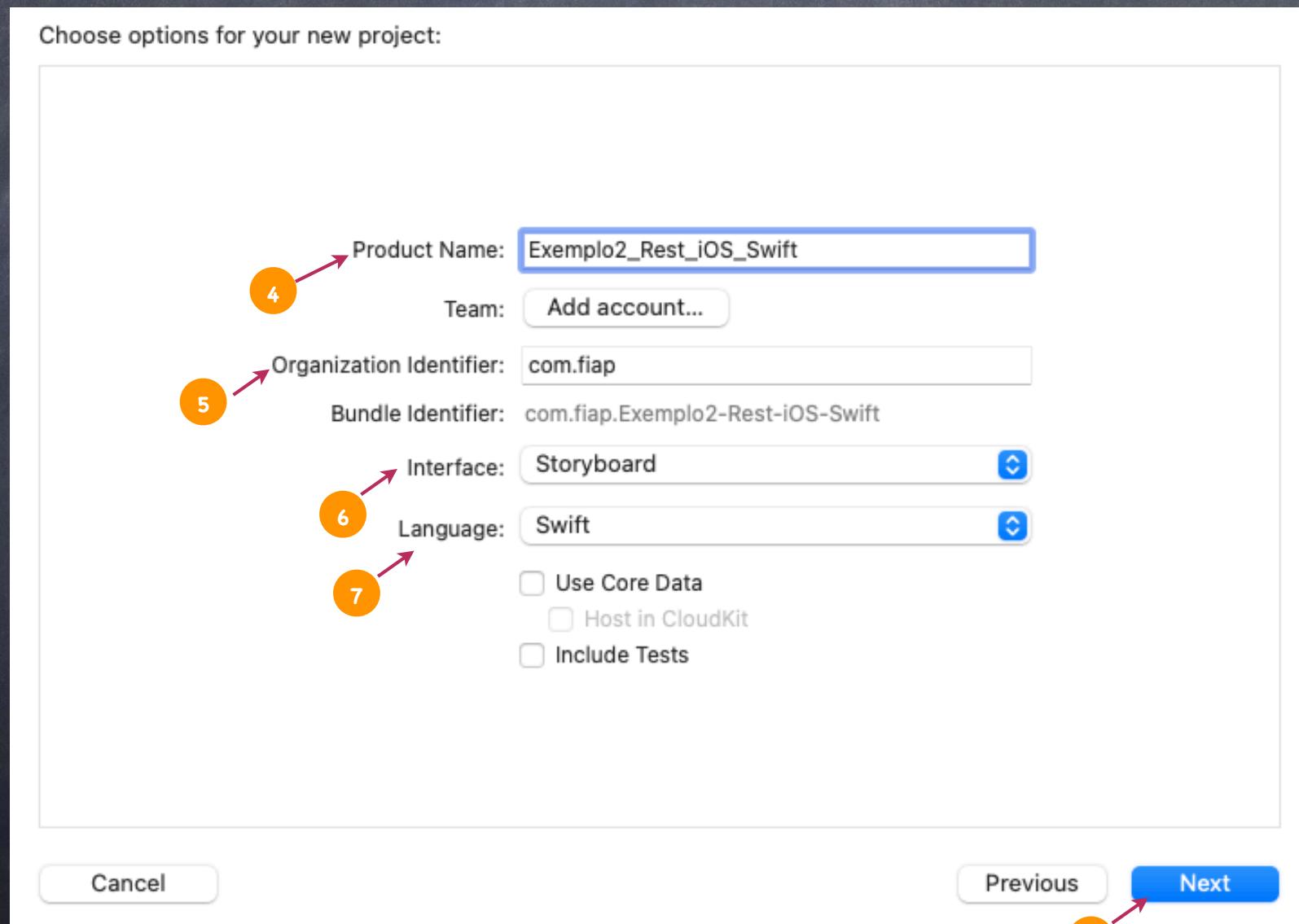
Iniciando um Projeto Mais Complexo

- Clique em File -> New Project -> IOS -> App -> Next



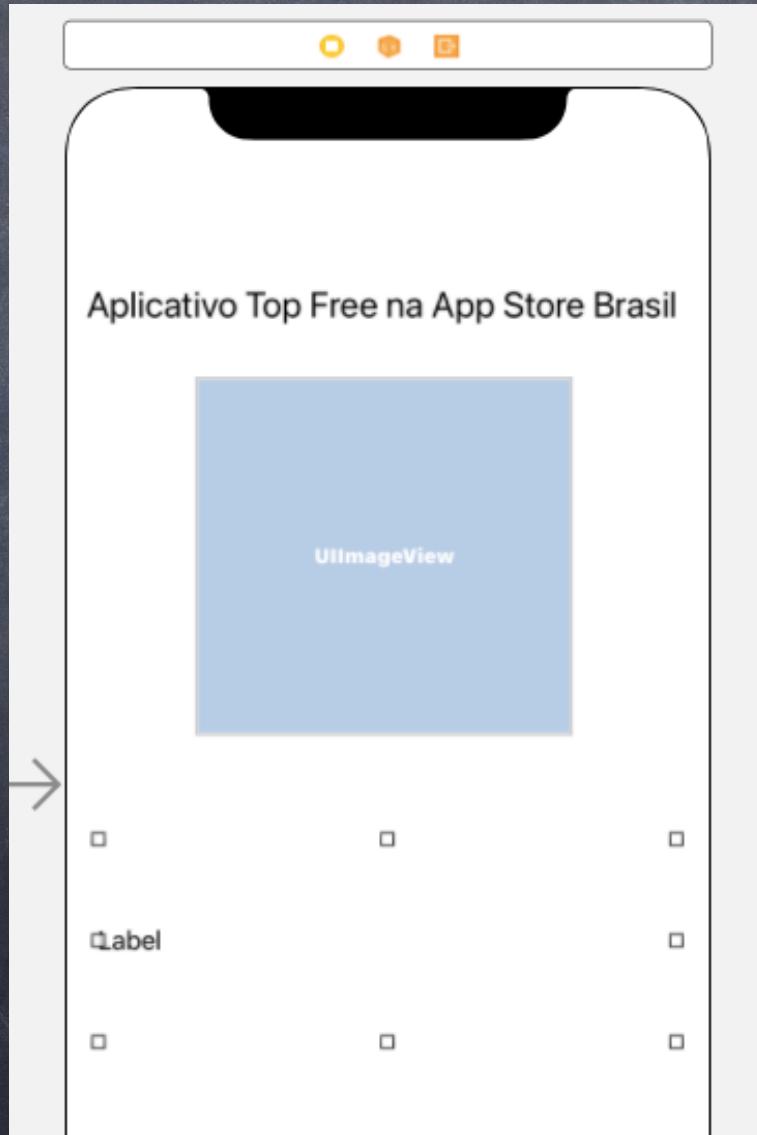
Os dados do Projeto

- Preencha com os dados abaixo, lembre-se de informar uma URL invertida no o Organization Identifier , em Interface escolha Storyboard, em language escolha Swift.



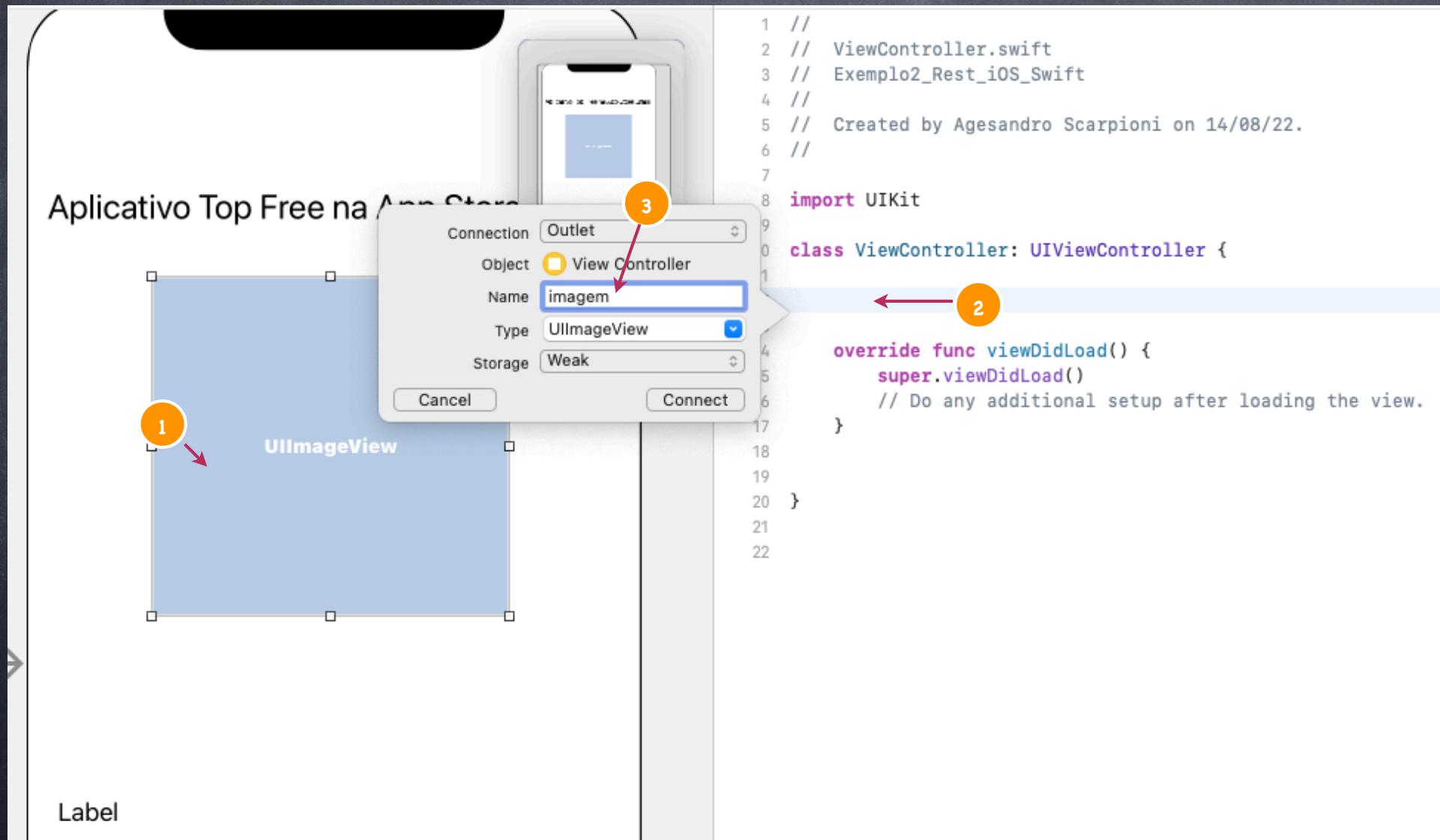
A Interface

- Desenhe a tela abaixo com 2 label's e 1 ImageView.



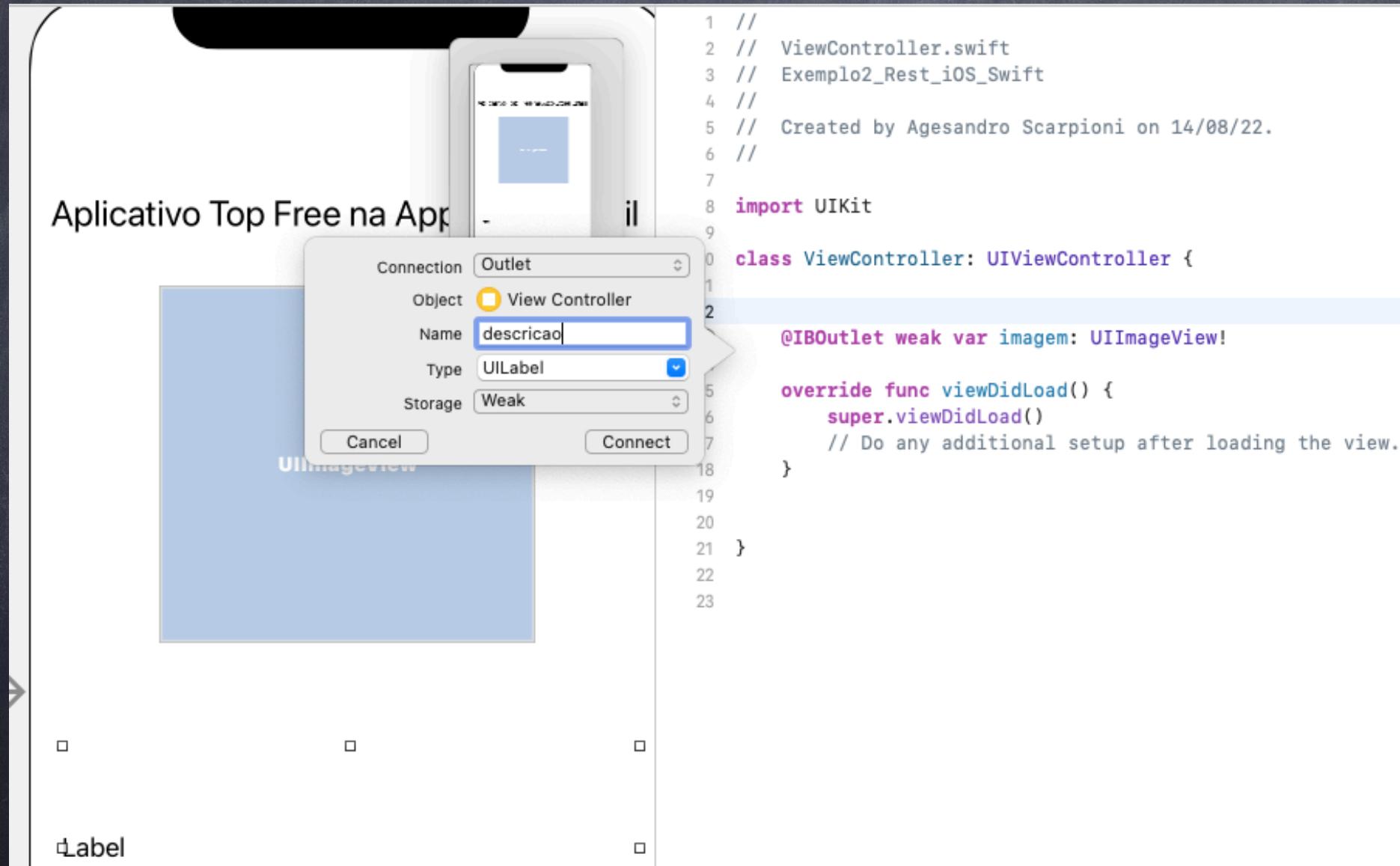
Definindo os IBOulet's

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



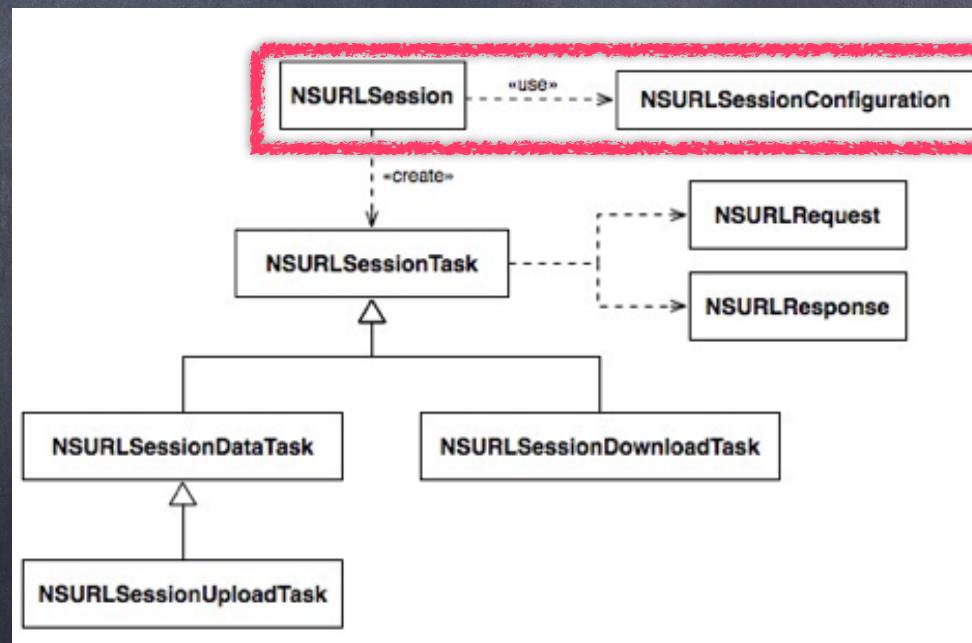
Definindo os IBOOutlet's

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

- ➊ 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

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

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;
```

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

Iniciando uma NSURLConnection

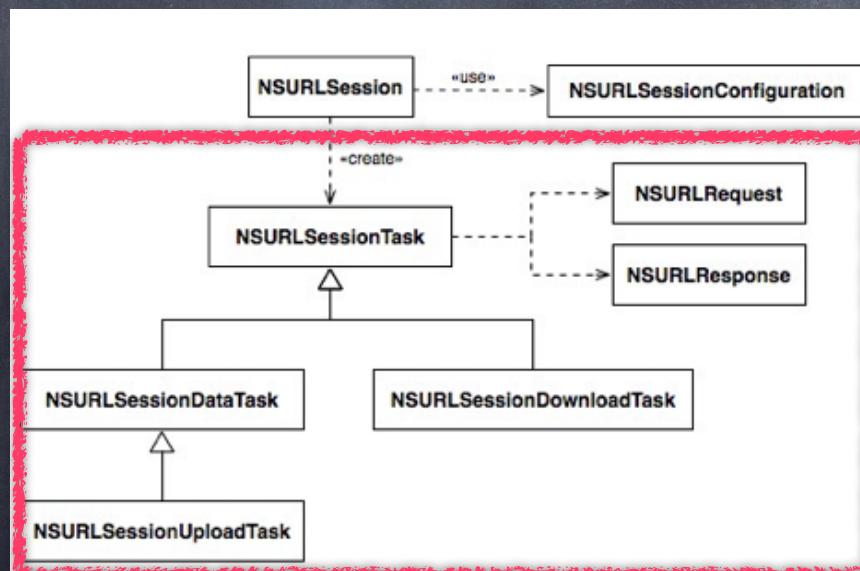
FIAP

- Implemente as linhas 16, 22 23 e 25 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 TableView que veremos em exercícios posteriores. Para facilitar copie a url daqui: <https://itunes.apple.com/br/rss/topfreeapplications/limit=1/json>

```
1 //  
2 // ViewController.swift  
3 // Exemplo2_Rest_iOS_Swift  
4 //  
5 // Created by Agesandro Scarpioni on 14/08/22.  
6 //  
7  
8 import UIKit  
9  
10 class ViewController: UIViewController {  
11  
12  
13     @IBOutlet weak var imagem: UIImageView!  
14     @IBOutlet weak var descricao: UILabel!  
15  
16     var session: URLSession?  
17  
18     override func viewDidLoad() {  
19         super.viewDidLoad()  
20         // Do any additional setup after loading the view.  
21         //Cria uma configuração de sessão default  
22         let sessionConfig = URLSessionConfiguration.default  
23         session = URLSession(configuration: sessionConfig)  
24         //URL de acesso a API do iTunes para retornar o APP Top Free  
25         let url = URL(string: "https://itunes.apple.com/br/rss/topfreeapplications/limit=1/json")  
26     }  
27  
28  
29 }
```

NSURLSessionTask

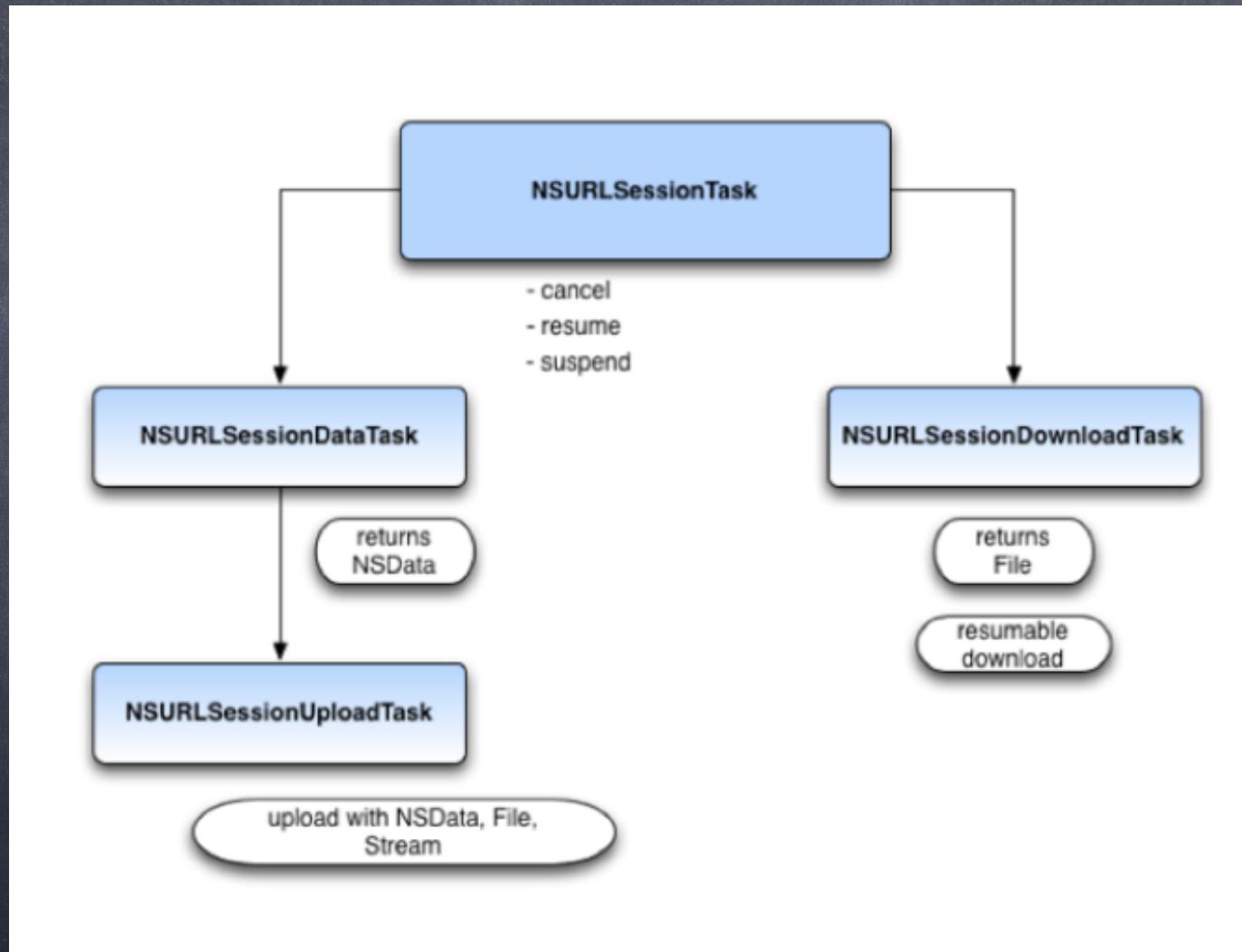
- Uma vez que a sessão é criada e configurada, pode se criar tarefas para serem executadas como download de arquivos e upload de dados. Tarefas são criadas através da classe `NSURLSessionTask`. Existem três tipos de tarefas possíveis:
 - `NSURLSessionDataTask`: criam requisições e recebem respostas dos 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

FIAP

Outras Características das Tasks



NSURLSessionTask

- Uma das maneiras de se criar uma task (NSURLSessionDataTask) é através do método `dataTask` 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 26 a 30.

```
25     let url = URL(string: "https://itunes.apple.com/br/rss/topfreeapplications/limit=1/json")
26     let task = session?.dataTask(with: url!, completionHandler: { data, response, error in
27         //ações que serão efetuadas quando a execução da task se completa
28     })
29     //disparo da execução da task
30     task?.resume()
31 }
```

- 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:
 - data**: `NSData`: buffer que recebe os dados em bytes. Precisa ser convertido para ser trabalhado em outros formatos

NSURLSessionTask

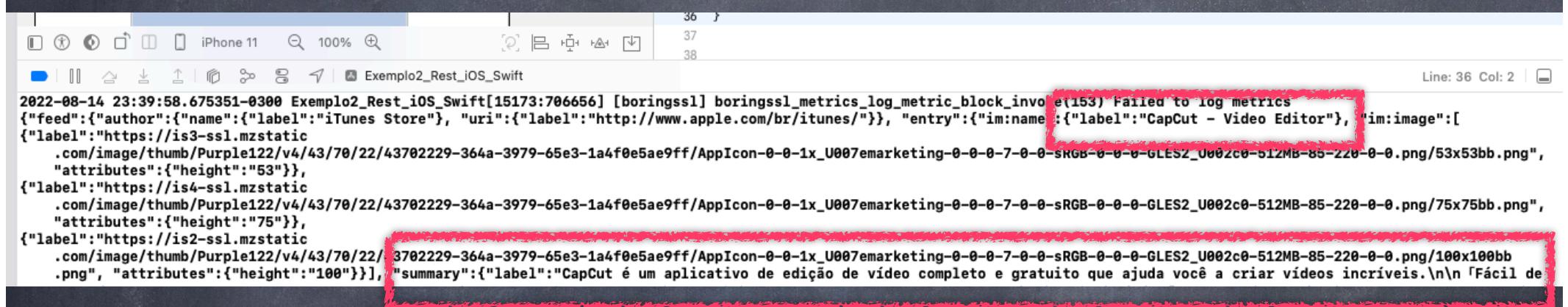
FIAP

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 28 e 29 para exibir o Json no console:

```
18     override func viewDidLoad() {  
19         super.viewDidLoad()  
20         // Do any additional setup after loading the view.  
21         //Cria uma configuração de sessão default  
22         let sessionConfig = URLSessionConfiguration.default  
23         session = URLSession(configuration: sessionConfig)  
24         //URL de acesso a API do iTunes para retornar o APP Top Free  
25         let url = URL(string: "https://itunes.apple.com/br/rss/topfreeapplications/limit=1/json")  
26         let task = session?.dataTask(with: url!, completionHandler: { data, response, error in  
27             //ações que serão efetuadas quando a execução da task se completa  
28             let texto = NSString(data: data!, encoding: String.Encoding.utf8.rawValue)  
29             print(texto!)  
30         })  
31         //disparo da execução da task  
32         task?.resume()  
33     }
```

Resultado no Console

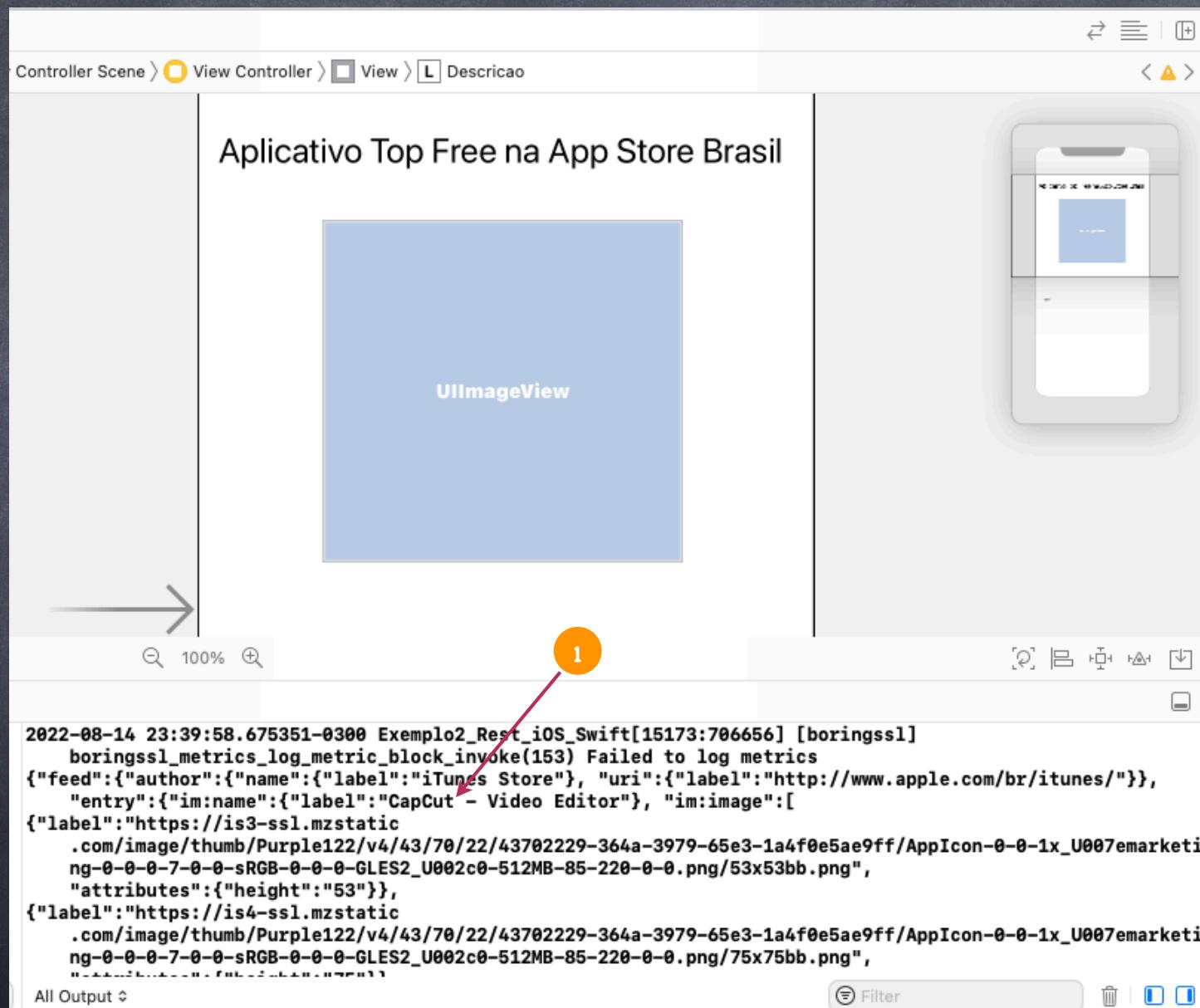
FIAP



```
iPhone 11 100% Exemplo2_Rest_iOS_Swift Line: 36 Col: 2
36 }
37
38
2022-08-14 23:39:58.675351-0300 Exemplo2_Rest_iOS_Swift[15173:706656] [boringssl] boringssl_metrics_log_metric_block_invoke(153) Failed to log metrics
{"feed": {"author": {"name": {"label": "iTunes Store"}, "uri": {"label": "http://www.apple.com/br/itunes/"}, "entry": {"im:name": {"label": "CapCut - Video Editor"}, "im:image": [{"label": "https://is3-ssl.mzstatic.com/image/thumb/Purple122/v4/43/70/22/43702229-364a-3979-65e3-1a4f0e5ae9ff/AppIcon-0-0-1x_U007emarketing-0-0-0-7-0-0-sRGB-0-0-0-GLES2_U002c0-512MB-85-220-0-0.png/53x53bb.png", "attributes": {"height": "53"}}, {"label": "https://is4-ssl.mzstatic.com/image/thumb/Purple122/v4/43/70/22/43702229-364a-3979-65e3-1a4f0e5ae9ff/AppIcon-0-0-1x_U007emarketing-0-0-0-7-0-0-sRGB-0-0-0-GLES2_U002c0-512MB-85-220-0-0.png/75x75bb.png", "attributes": {"height": "75"}}, {"label": "https://is2-ssl.mzstatic.com/image/thumb/Purple122/v4/43/70/22/43702229-364a-3979-65e3-1a4f0e5ae9ff/AppIcon-0-0-1x_U007emarketing-0-0-0-7-0-0-sRGB-0-0-0-GLES2_U002c0-512MB-85-220-0-0.png/100x100bb.png", "attributes": {"height": "100"}}], "summary": {"label": "CapCut é um aplicativo de edição de vídeo completo e gratuito que ajuda você a criar vídeos incríveis.\n\nFácil de usar, o CapCut permite que você adicione efeitos, música e transições para tornar seus vídeos mais divertidos e criativos. Com uma interface intuitiva, você pode editar seus vídeos em poucos segundos. O aplicativo também oferece opções de exportação para diferentes formatos, permitindo que você compartilhe seus resultados com amigos e familiares. Baixe o CapCut gratuitamente e comece a editar seus vídeos hoje mesmo!"}}}
```

Executando o App

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



Estrutura do Json da App Store

FIAP

```
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" = [  
                {  
                    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";  
                },  
                {  
                    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";  
                }  
            ]  
        }  
    }  
}
```

Array

[0]

[1]

Dica: Use no seu navegador o Jsonlint para abrir sua url com essa estrutura <https://jsonlint.com> ou <http://jsonviewer.stack.hu>

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 = "0.0000";
        currency = USD;
    label = Obter;
);
"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\u00eavel em minutos com o aplicativo Uber \u2014 sem reservas ou filas de espera de t\u00e1xi. \n\nTodas as op\u00e7\u00f5es, desde a viagem de baixo custo at\u00e1 premium, funcionam como uma atualiza\u00e7\u00e3o aos sistemas comuns. \n\nCrie sua conta para explorar o aplicativo. Adicione um cart\u00e3o de cr\u00e9ditos ou vincule seu PayPal, que sua tarifa ser\u00e1 cobrada automaticamente no final da sua viagem. Voc\u00e9 tamb\u00e9m pode pagar com dinheiro em algumas cidades. Depois de sua viagem, vamos enviar um recibo a voc\u00e9 por e-mail. \n\nVej se o Uber est\u00e1 dispon\u00e1vel 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\u00e1vidas? Acesse https://help.uber.com/";
};
```

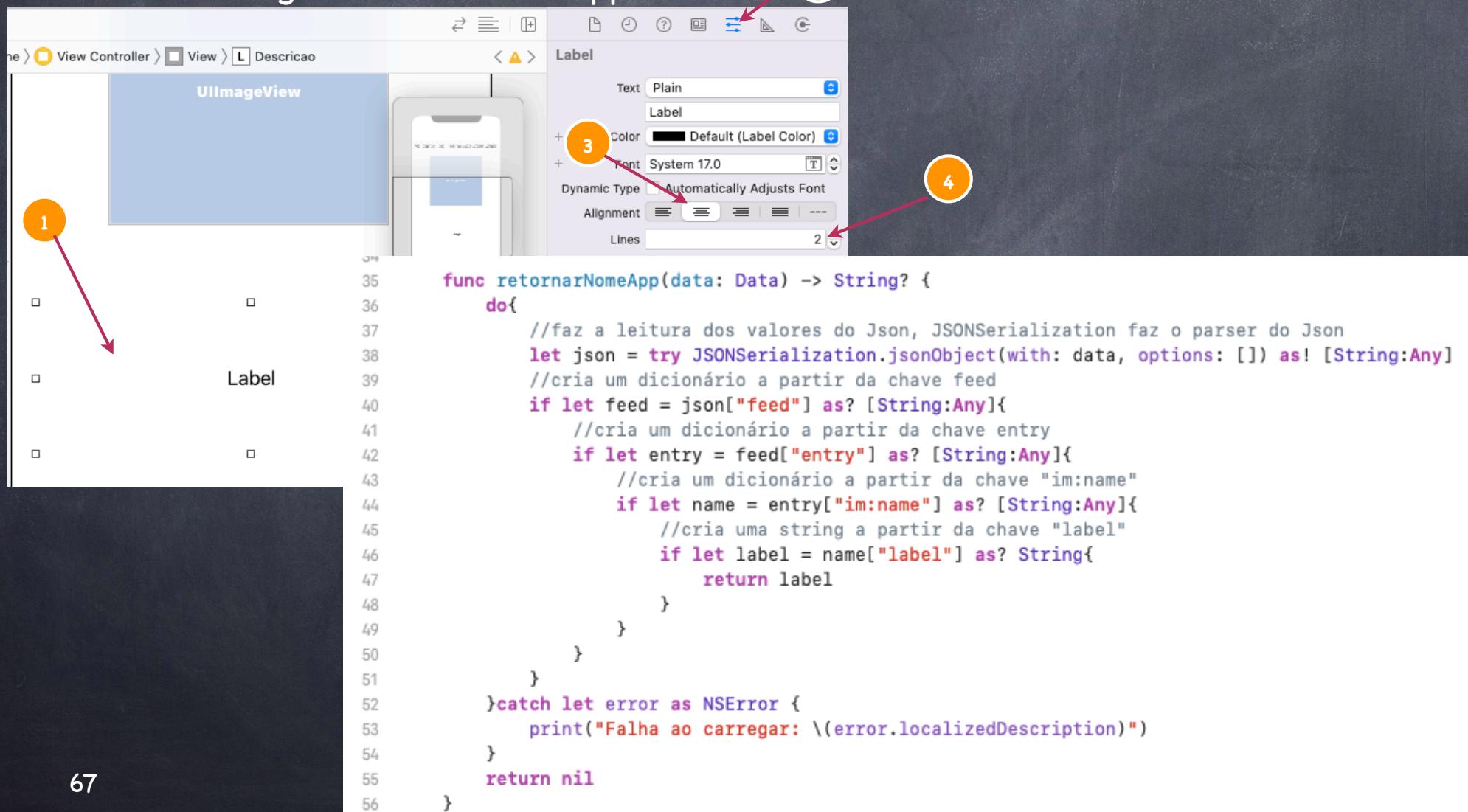
Continuação

```
        label = "\u000a9 Uber Technologies Inc.";
    };
    summary = {
        label = "Consiga uma viagem confi\u00eavel em minutos com o aplicativo Uber \u2014 sem reservas ou filas de espera de t\u00e1xi.\n\nTodas as op\u00e7\u00f5es, desde a viagem de baixo custo at\u00e1 premium, funcionam como uma atualiza\u00e7\u00e3o dos 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\u00e1 cobrada automaticamente no final da sua viagem. Voc\u00e9 tamb\u00f5m pode pagar com dinheiro em algumas cidades. Depois de sua viagem, vamos enviar um recibo a voc\u00e9 por e-mail.\n\nVeja se o Uber est\u00e1 dispon\u00e1vel 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\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\u00e1tis";
    };
    updated = {
        label = "2016-07-11T08:39:55-07:00";
    };
}
```

Exibir o nome Top Free

FIAP

- 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.



Exibir o nome Top Free

FIAP

- Depois de criar a função retornarNomeApp, implemente as linhas 30 até 34 para descarregar o nome que retorna da função para a constante "nomeApp", o label será atualizado na thread principal.

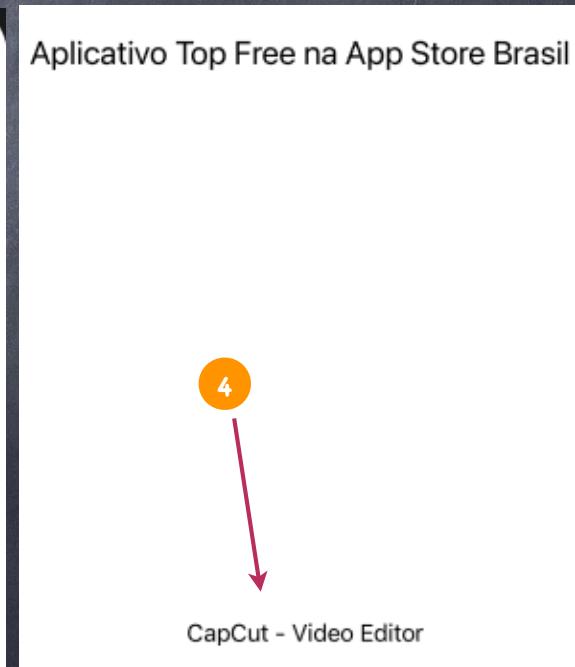
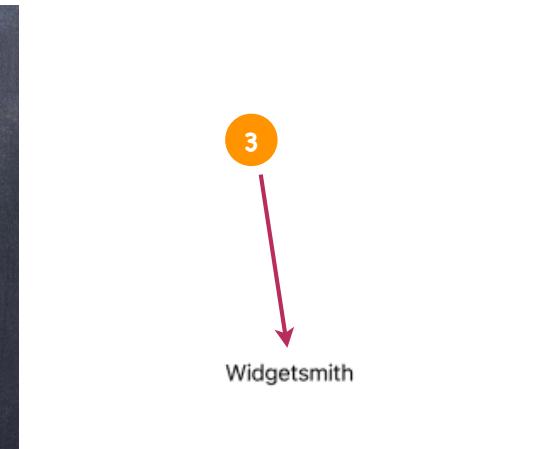
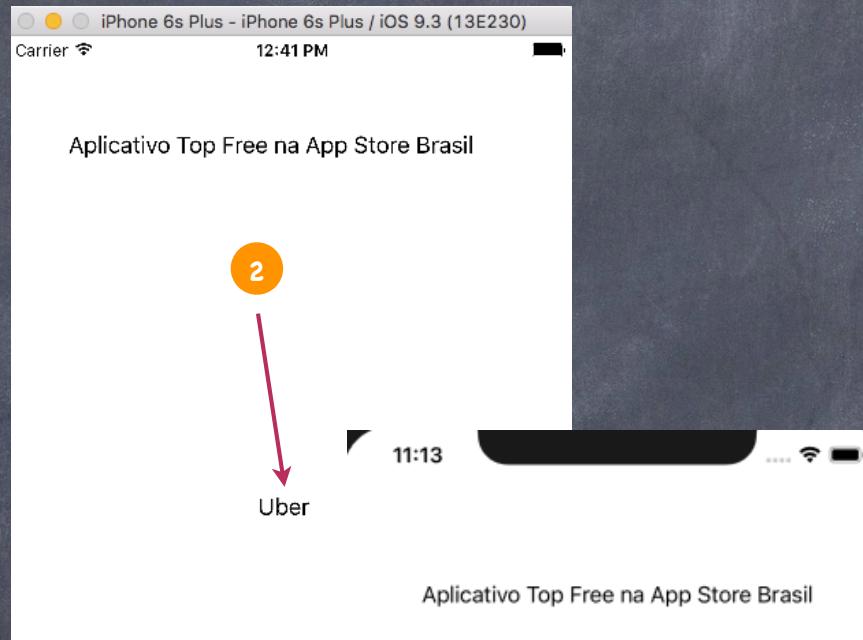
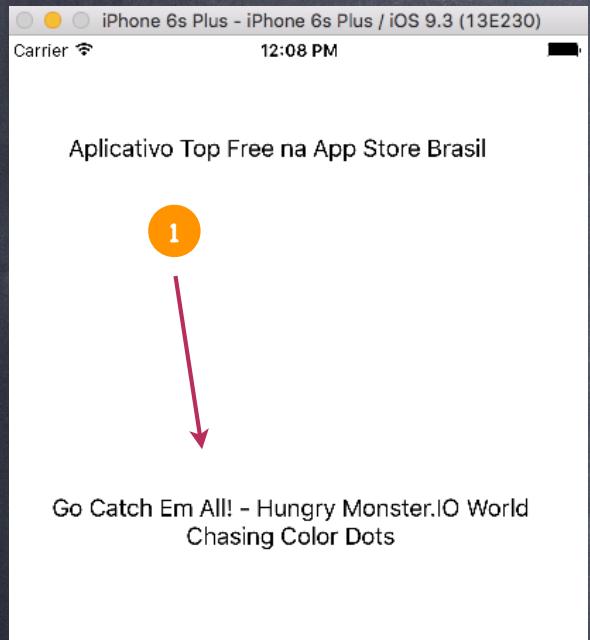
```
24 //URL de acesso à API do iTunes para localizar o top free
25 let url = URL(string: "https://itunes.apple.com/br/rss/topfreeapplications/limit=1/json")
26 let task = session?.dataTask(with: url!, completionHandler: { data, response, error in
27 //ações que serão efetuadas quando a execução da task se completa
28     let texto = NSString(data: data!, encoding: String.Encoding.utf8.rawValue)
29     print(texto!)
30     if let nomeApp = self.retornarNomeApp(data: data!){
31         DispatchQueue.main.async {
32             self.descricao.text = nomeApp
33         }
34     }
35 })
36 //disparo da execução da task
37 task?.resume()
38 }
```

Dica: Saiba mais para se trabalhar com thread em: https://developer.apple.com/library/mac/documentation/Performance/Reference/GCD_libdispatch_Ref/index.html

Resultado

FIAP

- 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), uma outra imagem em 26/09/2020 às 11h13 (3) e um mais atual em 14/08/2022 às 23h55h (4).



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.

```
2016-07-11 12:39:58.006 Exemplo2_Rest_iOS[2925:171158] {
    {
        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";
    },
    {
        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";
    },
    {
        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";
    }
}
```

1 → [0]

[1]

[2]

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.

```
64     func retornarImagenApp(data: Data) -> String?{
65         do {
66             //faz a leitura dos valores do Json, JSONSerialization faz o parser do Json
67             let json = try JSONSerialization.jsonObject(with: data, options: []) as! [String: Any]
68             //cria um dicionário a partir da chave feed
69             if let feed = json["feed"] as? [String:Any]{
70                 //cria um dicionário a partir da chave entry
71                 if let entry = feed["entry"] as? [String:Any]{
72                     //cria um array a partir da chave "im:image"
73                     if let name = entry["im:image"] as? [Any]{
74                         //cria um dicionário para receber a url da imagem na terceira posição do array
75                         if let image = name[2] as? [String:Any]{
76                             //cria uma string a partir da chave "label"
77                             if let label = image["label"] as? String{
78                                 return label
79                             }
80                         }
81                     }
82                 }
83             }
84         } catch let error as NSError {
85             print ("Falha ao carregar: \(error.localizedDescription)")
86         }
87         return nil
88     }
```

Exibindo a imagem

- Implemente uma nova função chamada carregarImagendeURL, 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.

```
72
93     func carregarImagenURL(imagemURL:String){
94         //cria uma URL da String imagemURL
95         let myUrl = URL(string: imagemURL)
96         let url = URLRequest(url: myUrl!)
97         //cria uma task do tipo Download
98         let session = URLSession.shared
99         let task = session.dataTask(with: url) { (data, response, error) in
100             //se resposta = not null recebe o binário da imagem
101             if let imageData = data{
102                 //transforma o binário em UIImage e atualiza a tela na thread principal
103                 DispatchQueue.main.async {
104                     self.imagem.image = UIImage(data: imageData)
105                 }
106             }
107         }
108     }
109     task.resume()
110 }
```

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 36 a 38.

```
26     let task = session?.dataTask(with: url!, completionHandler: { (data, response, error) in
27         //ações que serão efetuadas quando a execução da task se completa
28         let texto = NSString(data: data!, encoding: String.Encoding.utf8.rawValue)
29         print(texto!)
30
31         if let nomeApp = self.retornarNomeApp(data: data!){
32             DispatchQueue.main.async {
33                 self.descricao.text = nomeApp
34             }
35         }
36         if let appImageURL = self.retornarImagenApp(data: data!){
37             self.carregarImagenURL(imagenURL: appImageURL)
38         }
39     })
40     //disparo da execução da task
41     task?.resume()
42 }
```

Exibindo a imagem

FIAP

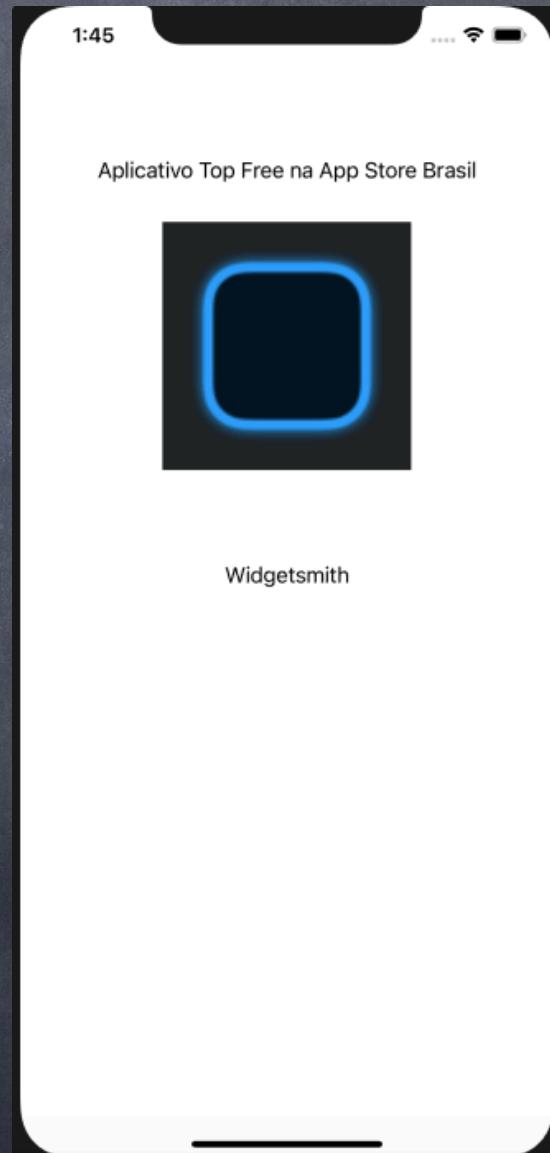
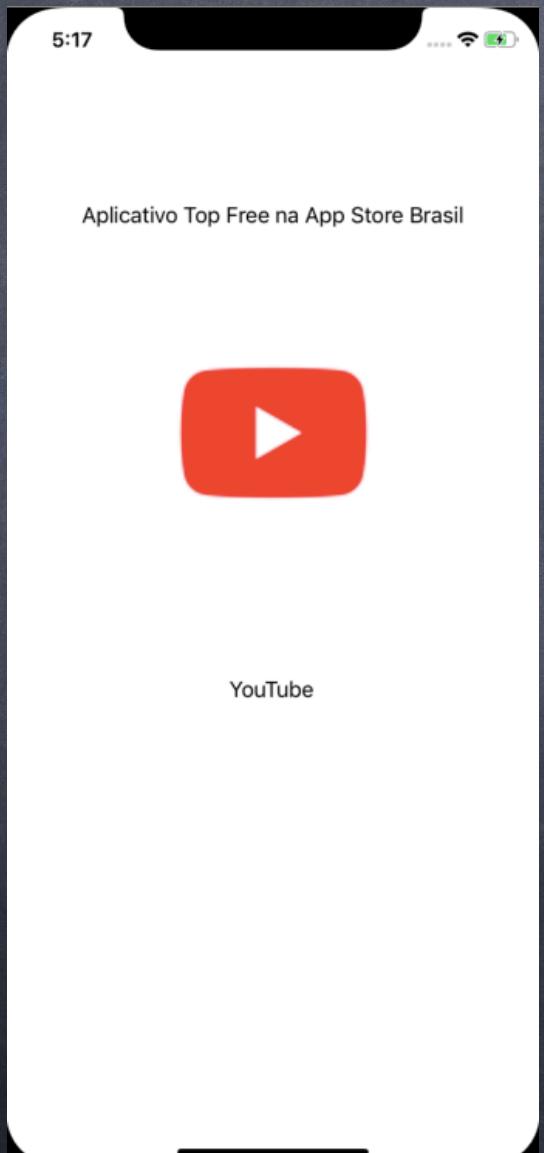
- ④ Observe as imagens do App top Free em 14/08/22 às 23:49



Exibindo a imagem

FIAP

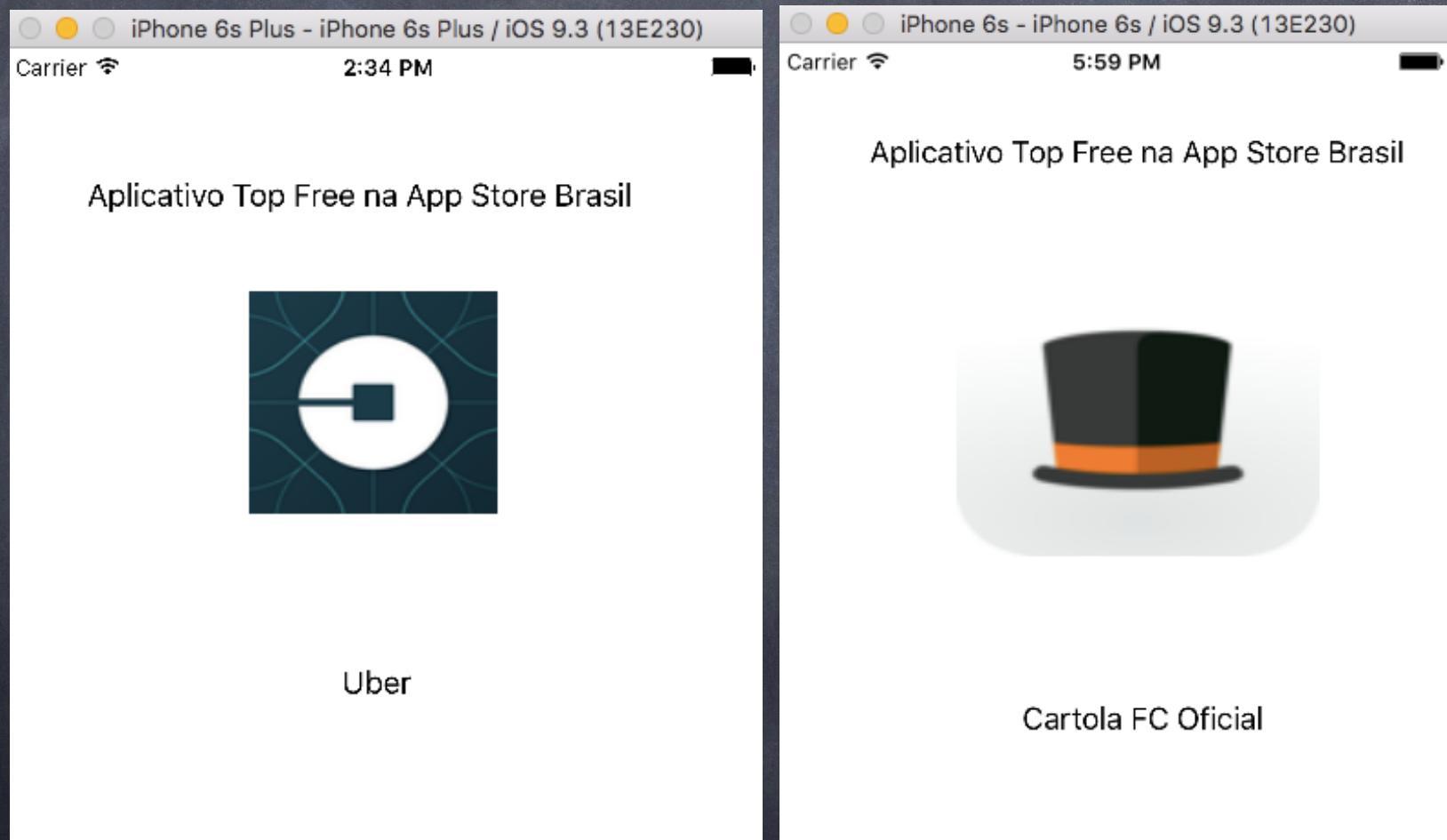
- Observe as imagens do App top Free em 08/09/19 às 5:17 PM e em 26/09/2020 1:45 PM



Exibindo a imagem

FIAP

- 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



Melhorias para o exemplo Raiz

- Se você quiser saber como seria possível melhorar a estrutura do Exemplo1_Rest_iOS_Swift, modificando o método para receber a chave como parâmetro as próximas imagens mostra a solução:

```
49
50     func retornar(data:Data, chave:String) -> String?{
51         var resposta:String?=nil
52         do{
53             //A linha abaixo faz a leitura dos valores do Json, NSJSONSerialization faz o parser do Json
54             let json = try JSONSerialization.jsonObject(with: data, options: []) as! [String:Any]
55             //cria e popula uma string a partir da chave neste exemplo "titulo"
56             if let retorno = json[chave] as? String{
57                 resposta = retorno
58             }
59         }catch let error as NSError{
60             resposta = "Falha ao carregar \(error.localizedDescription)"
61         }
62         return resposta
63     }
```

- Com um método mais genérico onde além de receber o Data, o método também recebe a chave, assim não precisamos ter um método para cada chave.

Melhorias para o exemplo Raiz

FIAP

- Chamando o método separadamente, assim se errarmos a chave, só a chave errada não será exibida, obviamente porque cada chamada está separada.

```
24
25 @IBAction func exibir(_ sender: Any) {
26     //cria uma configuração de sessão default
27     let config = URLSessionConfiguration.default
28     //cria uma sessão com a configuração default
29     session = URLSession(configuration: config)
30     //acesso a API
31     let url = URL(string: "https://xkcd.com/info.0.json")
32
33     let task = session?.dataTask(with: url!, completionHandler: { data, response, error in
34         //ações que serão efetuadas quando a execução do task se completa
35         //let texto = NSString(data: data!, encoding: String.Encoding.utf8.rawValue)
36         //print(texto!)
37         if let texto = self.retornar(data: data!, chave: "title"){
38             DispatchQueue.main.sync {
39                 self.titulo.text = texto
40             }
41         }
42
43         if let appImageURL = self.retornar(data: data!, chave: "img"){
44             DispatchQueue.main.sync {
45                 self.carregarImagenURL(imagemURL: appImageURL)
46             }
47         }
48     })
49     //disparo da execução do task
50     task?.resume()
51 }
```

Melhorias para o exemplo Raiz

FIAP

- Neste exemplo a chamada foi encadeada, ou seja, se errar uma das chaves, nada será exibido, no entanto, o código fica mais limpo.

```
24
25  @IBAction func exibir(_ sender: Any) {
26      //cria uma configuração de sessão default
27      let config = URLSessionConfiguration.default
28      //cria uma sessão com a configuração default
29      session = URLSession(configuration: config)
30      //acesso a API
31      let url = URL(string: "https://xkcd.com/info.0.json")
32
33      let task = session?.dataTask(with: url!, completionHandler: { data, response, error in
34          //ações que serão efetuadas quando a execução do task se completa
35          //let texto = NSString(data: data!, encoding: String.Encoding.utf8.rawValue)
36          //print(texto!)
37          if let texto = self.retornar(data: data!, chave: "title"){
38              if let appImageURL = self.retornar(data: data!, chave: "img"){
39                  DispatchQueue.main.sync {
40                      self.titulo.text = texto
41                      self.carregarImagemURL(imagemURL: appImageURL)
42                  }
43              }
44          }
45      })
46      //disparo da execução do task
47      task?.resume()
48  }
49
```

Melhorias para o exemplo Raiz

- Neste exemplo a chamada foi encadeada para chamar todas as chaves, note que para o ID o retorno é inteiro.

```
31     let url = URL(string: "https://xkcd.com/info.0.json")
32
33     let task = session?.dataTask(with: url!, completionHandler: { data, response, error in
34         //ações que serão efetuadas quando a execução do task se completa
35         //let texto = NSString(data: data!, encoding: String.Encoding.utf8.rawValue)
36         //print(texto!)
37         if let texto = self.retornar(data: data!, chave: "title"){
38             if let appImageURL = self.retornar(data: data!, chave: "img"){
39                 if let mes = self.retornar(data: data!, chave: "month"){
40                     if let ano = self.retornar(data: data!, chave: "year"){
41                         if let dia = self.retornar(data: data!, chave: "day"){
42                             if let numero = self.retornarInteiro(data: data!, chave: "num"){
43
44                                 DispatchQueue.main.sync {
45                                     self.titulo.text = texto
46                                     self.id.text = String(numero)
47                                     self.data.text = dia + "/" + mes + "/" + ano
48                                     self.carregarImagemURL(imagemURL: appImageURL)
49                                 }
50                             }
51                         }
52                     }
53                 }
54             }
55         })
56         //disparo da execução do task
57         task?.resume()
58     }
59 }
```

Melhorias para o exemplo Raiz

FIAP

- Aqui está a função que retorna o inteiro.

```
77     func retornarInteiro(data:Data, chave:String) -> Int?{
78         var resposta:Int?=nil
79         do{
80             //A linha abaixo faz a leitura dos valores do Json, NSJSONSerialization faz o parser do Json
81             let json = try JSONSerialization.jsonObject(with: data, options: []) as! [String:Any]
82             if let retorno = json[chave] as? Int{
83                 resposta = retorno
84             }
85         }catch let error as NSError{
86             print("Falha ao carregar \(error.localizedDescription)")
87         }
88         return resposta
89     }
90 }
```

Atividade

- Se você chegou até aqui pelo jeito “Raiz”, use esse endereço da loja de aplicativos da Apple para construir um novo projeto, no entanto, tente fazer do jeito “Nutella” apresentado na página 28, ou seja, monte um struct para receber os dados, veja como na próxima página.
- Essa atividade é importantíssima para que consiga preparar um struct apropriado para cada retorno de Json.

Struct para Json complexo 2

Faça isso quando a chave do dicionário tem ":" no nome ou algum outro caracter que prejudica a montagem do Struct

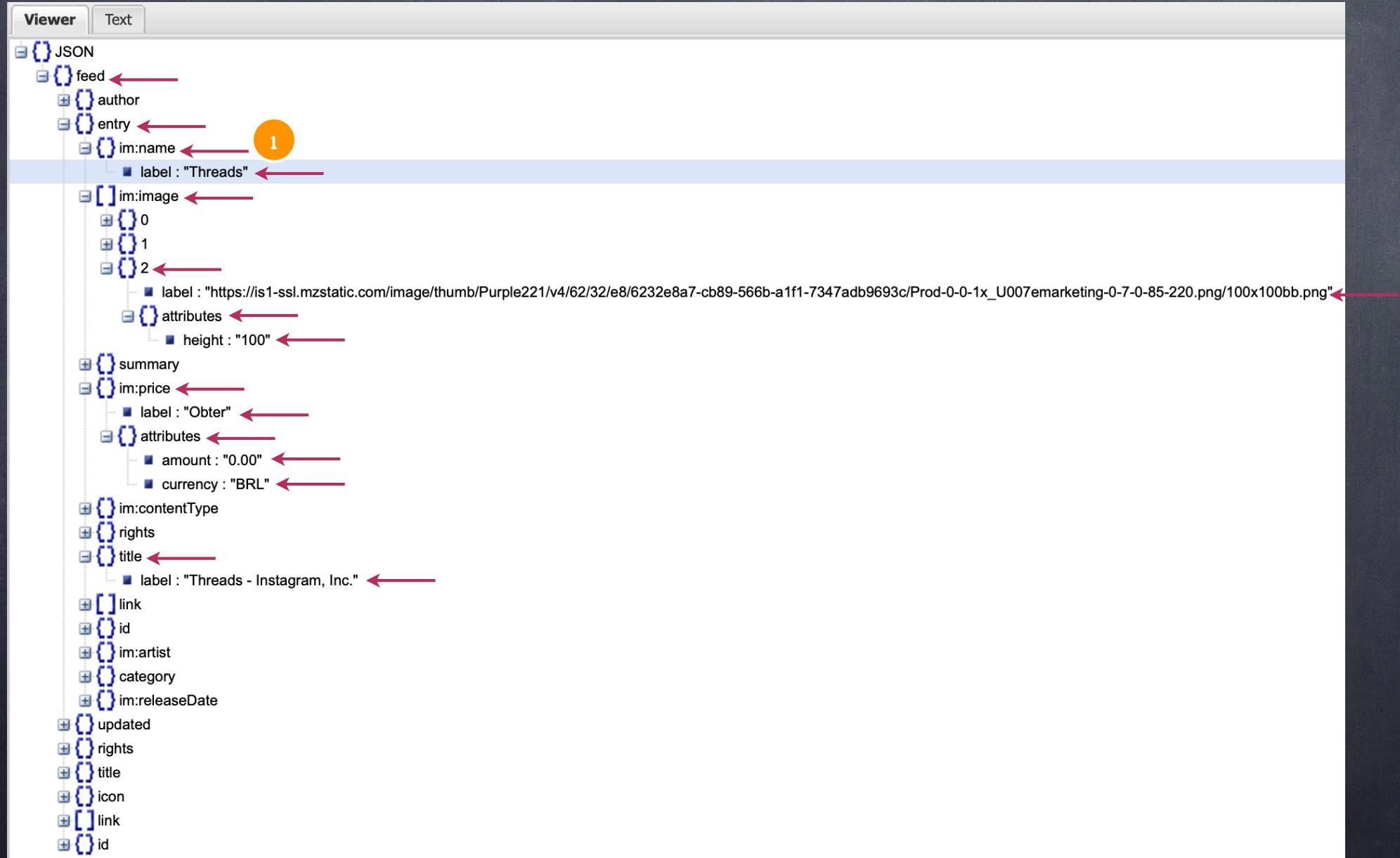
Veja a URL abaixo e coloque-a no JsonViewer (<https://jsonviewer.stack.hu/>)

<https://itunes.apple.com/br/rss/topfreeapplications/limit=1/json>

Struct para Json complexo 2

FIAP

Note que a chave do dicionário tem ":" (1) no nome

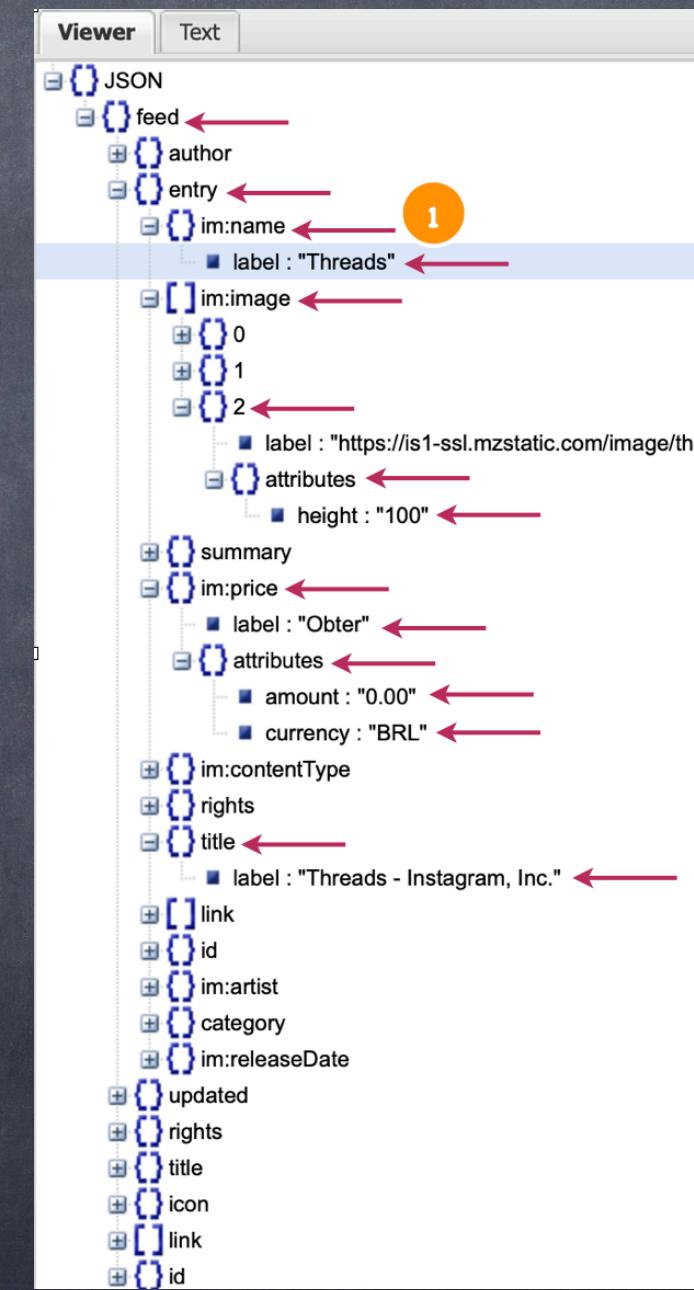


Struct para Json complexo 2

FIAP

Criando o Struct com CodingKeys

```
8 import Foundation
9 struct App: Decodable {
10     var feed:Feed
11 }
12 struct Feed: Decodable {
13     var entry:Entry
14 }
15 struct Entry: Decodable {
16     var title:Title
17     var name: Info
18     var image: [Info]
19     var price: Info
20     enum CodingKeys: String, CodingKey {
21         case name = "im:name"
22         case image = "im:image"
23         case price = "im:price"
24         case title
25     }
26 }
27 struct Info: Decodable {
28     var label: String
29     var attributes: Attributes?
30 }
31 struct Title: Decodable {
32     var label:String
33 }
34 struct Attributes: Decodable {
35     let amount: String?
36     let currency: String?
37     let height: String?
38 }
```



Struct para Json complexo 2

FIAP

Print no console dos resultados

```
func load(){
    let jsonUrlString = "https://itunes.apple.com/br/rss/topfreeapplications/limit=1/json"
    let url = URL(string: jsonUrlString)

    URLSession.shared.dataTask(with: url!) { data, response, error in

        guard let data = data else {return}
        do {
            app = try JSONDecoder().decode(App.self, from: data)

            print(app.feed.entry.title.label)
            print(app.feed.entry.name.label)
            print(app.feed.entry.image[0].label)
            print(app.feed.entry.price.label)
            print(app.feed.entry.price.attributes?.amount as Any) // ou
            print(app.feed.entry.price.attributes?.currency ?? "Sem dado")

        }catch let jsonError {
            print("Error serialization Json", jsonError)
        }
    }
    .resume()
}
```

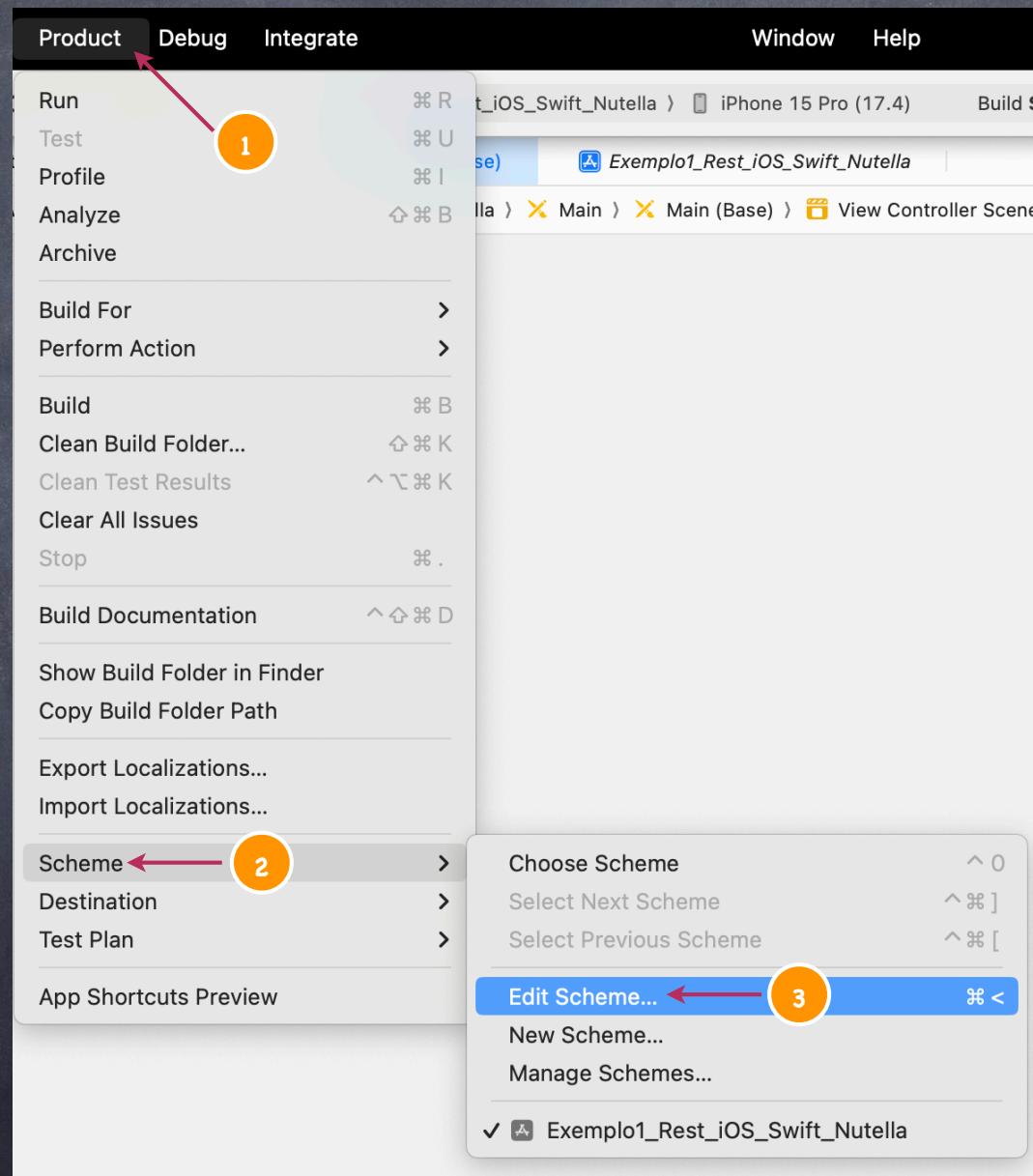
Struct para Json complexo 2

FIAP

Exibindo nos Label's os resultados

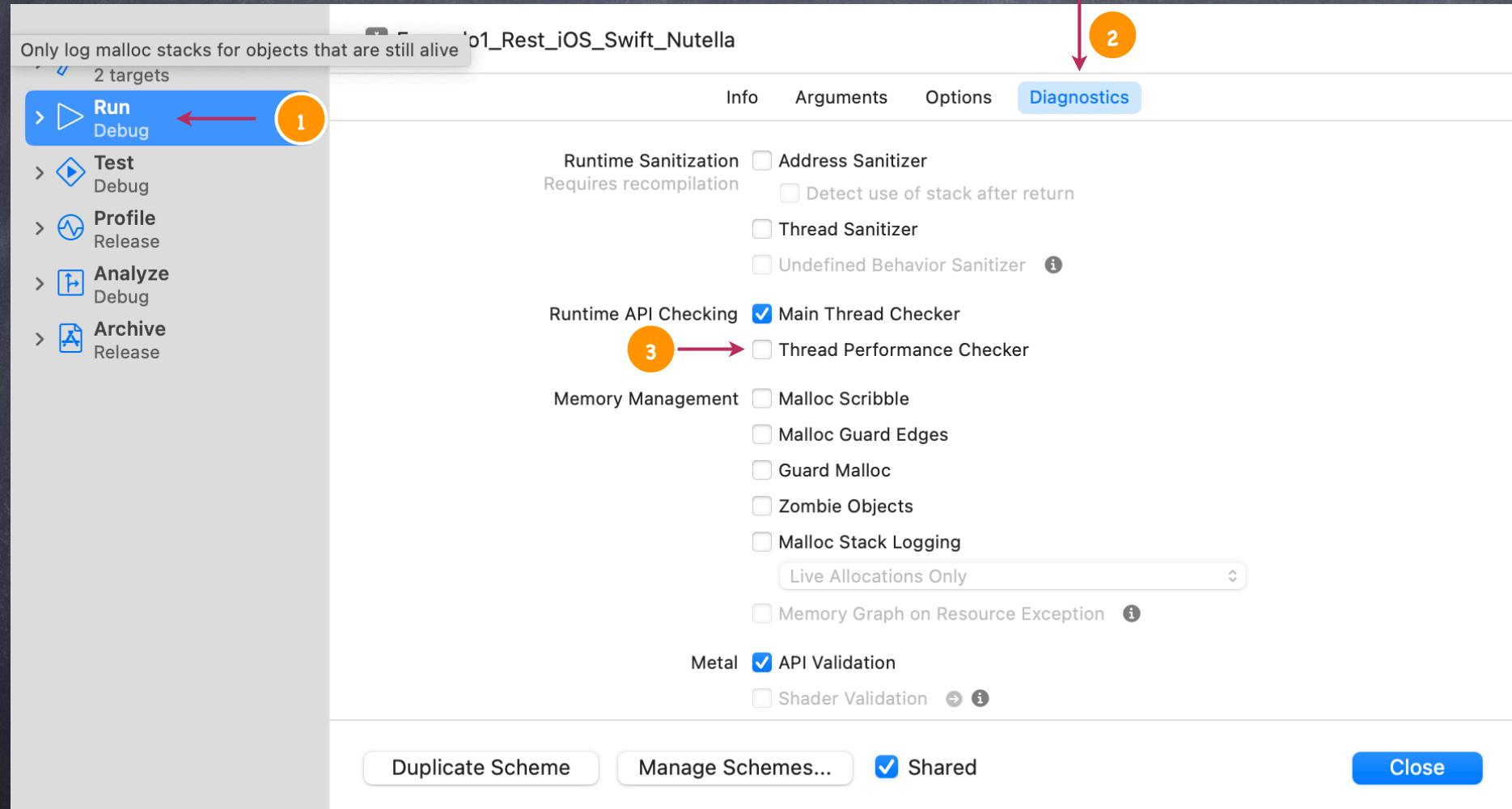
```
31 func load(){
32     let jsonUrlString = "https://itunes.apple.com/br/rss/topfreeapplications/limit=1/json"
33     let url = URL(string: jsonUrlString)
34
35     URLSession.shared.dataTask(with: url!) { data, response, error in
36         guard let data = data else {return}
37         do {
38             app = try JSONDecoder().decode(App.self, from: data)
39             //print(app.feed.entry.title.label)
40             //print(app.feed.entry.name.label)
41             //print(app.feed.entry.image[0].label)
42             //print(app.feed.entry.price.label)
43             //print(app.feed.entry.price.attributes?.amount as Any) // ou
44             //print(app.feed.entry.price.attributes?.currency ?? "Sem dado")
45             let imagem = self.carregarImagem(urlImagen: app.feed.entry.image[0].label)
46
47             DispatchQueue.main.sync {
48                 self.lblTitulo.text = app.feed.entry.name.label
49                 self.lblPreco.text = app.feed.entry.price.attributes?.amount
50                 self.lblMoeda.text = app.feed.entry.price.attributes?.currency
51                 self.minhaImagem.image = imagem
52             }
53
54         }catch let jsonError {
55             print("Error serialization Json", jsonError)
56         }
57     }
58     .resume()
59 }
```

Desligando o Thread Performance Checker



Dica: Saiba mais em: <https://developer.apple.com/documentation/xcode/diagnosing-performance-issues-early>

Desligando o Thread Performance Checker



Dica: Saiba mais em: <https://developer.apple.com/documentation/xcode/diagnosing-performance-issues-early>