

PROF. RENÊ XAVIER

---

# DESENVOLVIMENTO PARA IOS 11 COM SWIFT 4

ONDE ENCONTRAR O MATERIAL?

[HTTPS://GITHUB.COM/RENEFX/IOS-2018-01](https://github.com/RENEFX/IOS-2018-01)

---

**GITHUB**

ALÉM DO TRADICIONAL BLACKBOARD DO IESB

O QUE VAMOS FAZER HOJE?

---

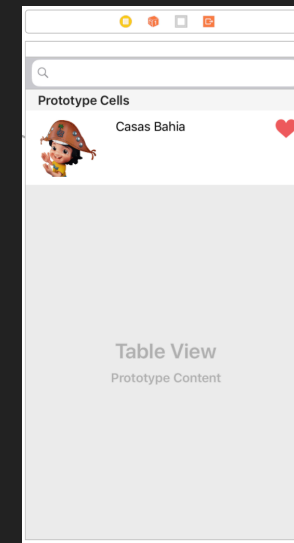
## AGENDA

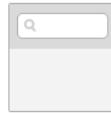
- ▶ Filter, Map, Reduce
- ▶ Search Bar
- ▶ Camada Model



JÁ AJUSTAMOS NOSSOS  
BAHIANINHOS.

AGORA,  
COMO PESQUISAR AS  
LOJAS NA TABLEVIEW?





**Search Bar** - Displays an editable search bar, containing the search icon, that sends an action message t...

---

# SEARCH BAR

TABLEVIEW CONTROLLER

---

**CALMA, VAMOS DOMINAR  
OPERAÇÕES COM ARRAYS  
PRIMEIRO**



---

**FILTER, MAP E  
REDUCE**

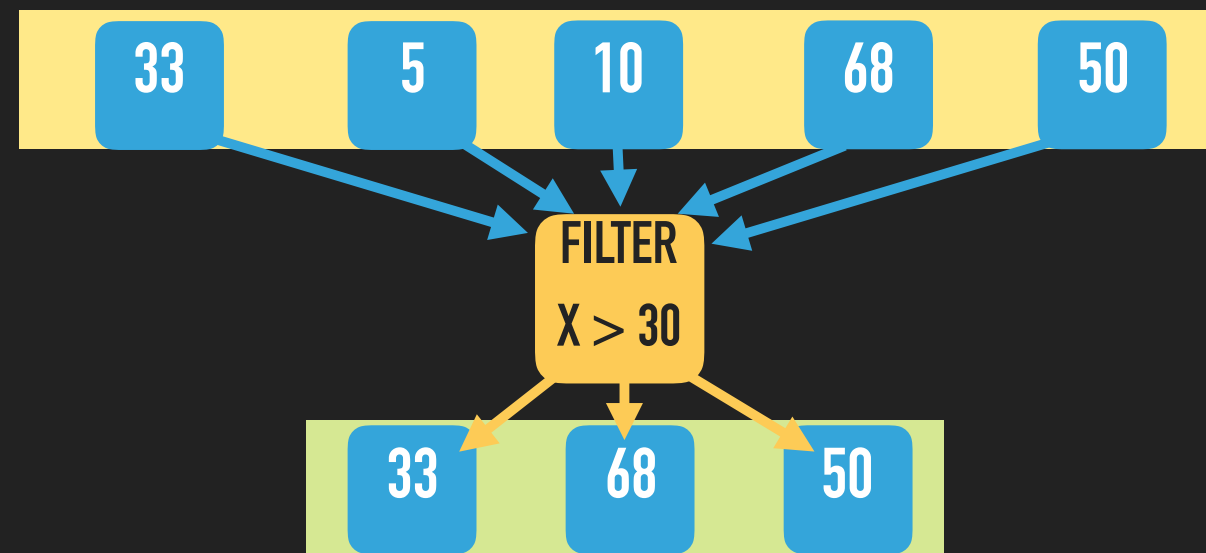
## FILTER, MAP E REDUCE

---

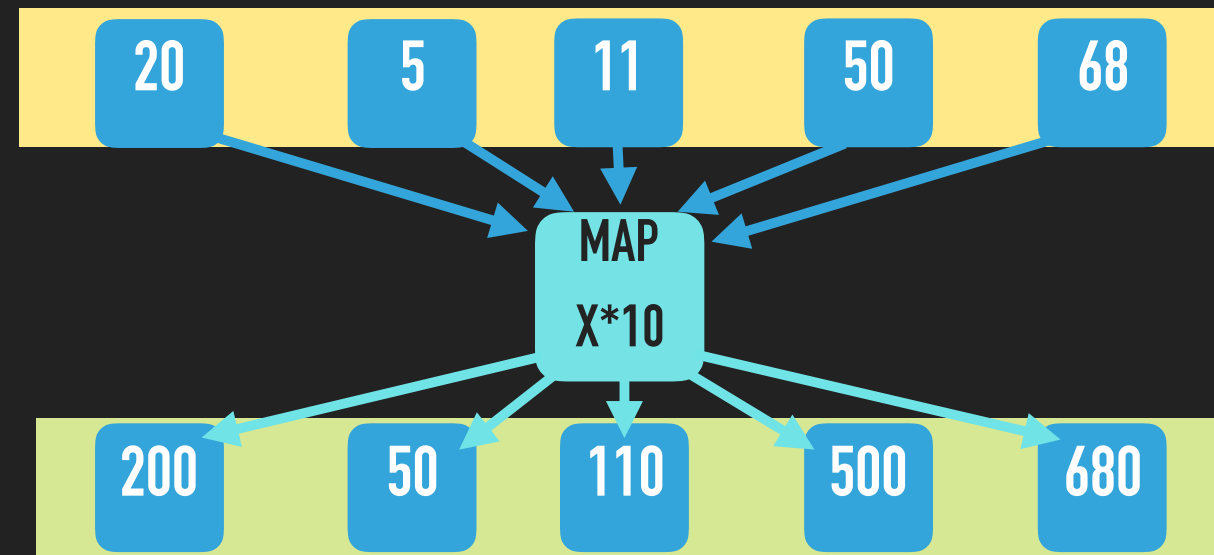
- ▶ São operações com arrays para evitar o uso de um **for**
- ▶ Filter - Retorna um array composto por elementos de acordo com as condições que você colocar
- ▶ Map - Executa uma operação em todos os elementos do array e então retorna um novo array com esses valores
- ▶ Reduce - Combina todos os valores de um array aplicando uma mesma operação. O retorno não é um array, mas um valor único resultante da combinação de cada um dos elementos



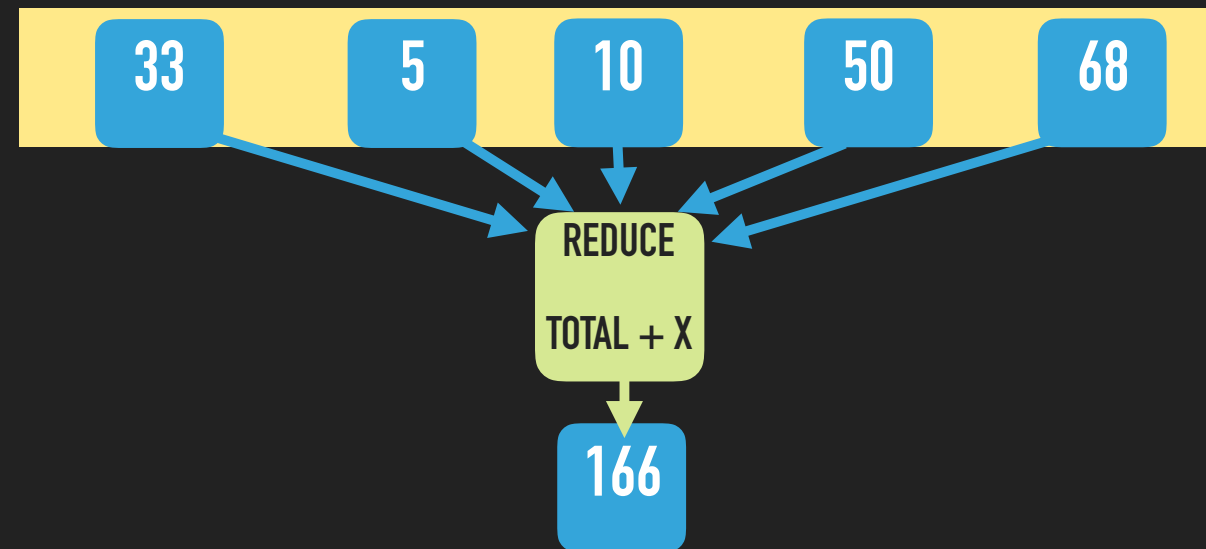
## FILTER



## MAP



## REDUCE



## FILTER, MAP E REDUCE

### FILTER

```
let arrayNumeros = [33, 5, 10, 68, 50]
```

```
arrayNumeros.filter(isIncluded: (Int) throws -> Bool)
```

```
arrayNumeros.filter { (Int) -> Bool in  
code  
}
```

```
arrayNumeros.filter { (numero) -> Bool in  
return numero > 30  
}
```

RETORNA UM ARRAY [33, 68, 50]

```
arrayNumeros.filter { (numero) -> Bool in return numero > 30 }
```

## FILTER, MAP E REDUCE

---

### FILTER

```
arrayNumeros.filter { (numero) -> Bool in return numero > 30 }
```

```
arrayNumeros.filter { numero -> Bool in return numero > 30 }
```

```
arrayNumeros.filter { numero in return numero > 30 }
```

```
arrayNumeros.filter { return $0 > 30 }
```

```
arrayNumeros.filter { $0 > 30 }
```

## MAP

```
let arrayNumeros = [33, 5, 10, 68, 50]
```

```
arrayNumeros.map(transform: (Int) throws -> T)
```

```
arrayNumeros.map({ numero in  
    return numero * 10  
})
```

RETORNA UM ARRAY

```
[330, 50, 100, 680, 500]
```

### MAP

```
let arrayNumeros = [33, 5, 10, 68, 50]
```

```
arrayNumeros.map(transform: (Int) throws -> T)
```

```
arrayNumeros.map({ numero in  
    if numero > 30 {  
        return "a"  
    } else {  
        return "b"  
    }  
})
```

```
arrayNumeros.map({ numero in  
    return numero > 30 ? "a" : "b"  
})
```

```
arrayNumeros.map { $0 > 30 ? "a" : "b" }
```

```
["a", "b", "b", "a", "a"]
```

## FILTER, MAP E REDUCE

### REDUCE

```
let arrayNumeros = [33, 5, 10, 68, 50]
```

```
let y = arrayNumeros.reduce(initialResult: Result, nextPartialResult: (Result, Int) throws -> Result)
```

```
let y = arrayNumeros.reduce(0) { (total, numero) -> Int in  
    return total + numero  
}
```

RETORNA UM VALOR

**166**



## FILTER, MAP E REDUCE

### REDUCE

```
let arrayNumeros = [33, 5, 10, 68, 50]
```

```
let y = arrayNumeros.reduce(initialResult: Result, nextPartialResult: (Result, Int) throws -> Result)
```

```
let y = arrayNumeros.reduce(0) { (total, numero) -> Int in  
    return total + numero  
}
```

RETORNA UM VALOR

**166**

## REDUCE

```
let y = arrayNumeros.reduce(0) { (total, numero) -> Int in  
    return total + numero  
}
```

```
arrayNumeros.reduce(0) { $0 + $1 }
```

```
arrayNumeros.reduce(0, +)
```

FILTER, MAP E REDUCE

# PODEMOS COMBINAR ESSAS FUNÇÕES

```
let arrayNumeros = [33, 5, 10, 68, 50]
```

```
arrayNumeros.filter({ $0 % 10 == 0 }).map({ $0 / 2 }).reduce(0) { $0 + $1 }
```

30

FILTER, MAP E REDUCE

---

**ESSAS FUNÇÕES NÃO  
SERVEM SÓ PARA  
NÚMEROS**

## FILTER, MAP E REDUCE

---

```
let arrayTexto = ["a", "b", "c"]  
arrayTexto.reduce("") { $0 + $1 }
```

```
arrayTexto.reduce("", +)
```

**abc**

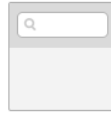
```
arrayTexto.map { $0 + ";" }
```

**["a;", "b;", "c;"]**

# VAMOS PARA O QUE INTERESSA

```
let arrayTexto = ["casa verde", "verdurão", "shopping"]  
arrayTexto.filter { $0.contains("verd") }
```

```
["casa verde", "verdurão"]
```



**Search Bar** - Displays an editable search bar, containing the search icon, that sends an action message t...

---

# SEARCH BAR

## SEARCH BAR

- ▶ No storyboard, adicione uma search bar



**Search Bar** - Displays an editable search bar, containing the search icon, that sends an action message t...



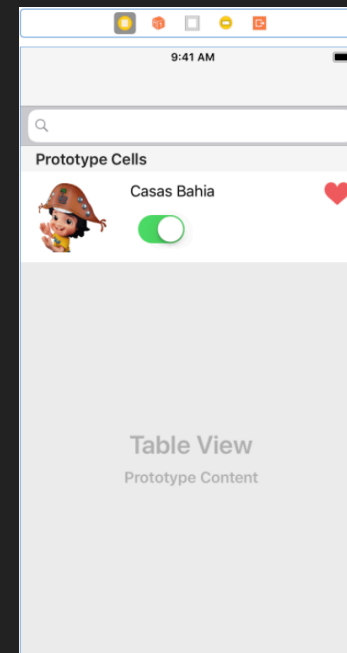
Table View



Search Bar



lojaCelulaIdentifier





# SINTO DIZER QUE É UM DELEGATE


- ▶ Declaramos o protocolo
- ▶ Informamos que implementamos o protocolo
- ▶ Implementamos o método do protocolo

```
class MinhaListaEstaticaTableViewController: UITableViewController, UISearchBarDelegate {
```


```
class ListaViewController: UIViewController, UITableViewDelegate, UITableViewDataSource, UISearchBarDelegate {
```

## SEARCH BAR

- ▶ Informamos que implementamos o protocolo
  - ▶ Temos que informar para a Search Bar que implementamos o procolo
  - ▶ Precisamos de um IBOutlet da Search Bar
  - ▶ Após isso preenchemos a property delegate



```
15
16  @IBOutlet weak var tableView: UITableView!
17  @IBOutlet weak var searchBar: UISearchBar!
18
19  override func viewDidLoad() {
20      super.viewDidLoad()
21
22      tableView.delegate = self
23      tableView.dataSource = self
24      searchBar.delegate = self
25
```



## SEARCH BAR

---

- ▶ Implementamos o método do protocolo
  - ▶ O método `textDidChange` informa quando o texto de busca mudou
  - ▶ `searchText` - O texto que acaba de ser digitado
  - ▶ Mesmo quando está apagando esse método informa
  - ▶ Temos que implementar uma busca

```
func searchBar(_ searchBar: UISearchBar, textDidChange searchText: String) {  
}
```

SEARCH BAR

---

# POR QUÊ? COMO IMPLEMENTAR A BUSCA?

## SEARCH BAR

---

```
func searchBar(_ searchBar: UISearchBar, textDidChange searchText: String) {  
    arrayNome = arrayNome.filter { $0.contains(searchText) }  
    tableView.reloadData()  
}
```

# POR QUE NÃO ESTÁ MUDANDO?

- ▶ Alteramos nosso array, mas não informamos à TableView que algo mudou
- ▶ Temos que recarregar o conteúdo dela com o reloadData()

```
func searchBar(_ searchBar: UISearchBar, textDidChange searchText: String) {  
    arrayNome = arrayNome.filter { $0.lowercased().range(of: searchText.lowercased()) != nil }  
    tableView.reloadData()  
}
```

## SEARCH BAR

---

```
func searchBar(_ searchBar: UISearchBar, textDidChange searchText: String) {  
    arrayNome = arrayNome.filter { $0.contains(searchText) }  
    tableView.reloadData()  
}
```

# TENTEM ENCONTRAR ERROS

- ▶ Apagar não retorna os dados anteriores
- ▶ Não consulta o meio das palavras
- ▶ Não funciona com maiúsculas/minúsculas diferentes


## SEARCH BAR

### 1. Apagar não retorna os dados anteriores

- ▶ Temos que criar outro array que não é nunca alterado. Assim, podemos sempre voltar para ele.
- ▶ Vamos substituir o uso do outro array por esse - `numberOfRowsInSection` & `cellForRowAt`
- ▶ Na `viewDidLoad` vamos preencher ele com nosso antigo array

```
var arrayFiltered:[String] = []
```

```
override func viewDidLoad() {  
    super.viewDidLoad()  
  
    tableView.delegate = self  
    tableView.dataSource = self  
    searchBar.delegate = self  
  
    arrayFiltered = arrayNome
```



## SEARCH BAR

---

### 2. Apagar não retorna os dados anteriores

- ▶ Na `textDidChange`, vamos informar que queremos o filtro do array original
- ▶ Caso o texto seja vazio (`""`), queremos todos os elementos originais

```
func searchBar(_ searchBar: UISearchBar, textDidChange searchText: String) {  
    if (searchText == "") {  
        arrayFiltered = arrayNome  
    } else {  
        arrayFiltered = arrayNome.filter { $0.contains(searchText) }  
    }  
    tableView.reloadData()  
}
```



## SEARCH BAR

---

### 2. Não consulta o meio das palavras

- ▶ O método `contains` busca somente no início das palavras
- ▶ Ao buscar o `range`, buscamos onde está aquela palavra na outra e para o caso não encontre a palavra na outra, o `range` é `nil`

```
func searchBar(_ searchBar: UISearchBar, textDidChange searchText: String) {  
    if (searchText == "") {  
        arrayFiltered = arrayNome  
    } else {  
        arrayFiltered = arrayNome.filter { $0.range(of: searchText) != nil }  
    }  
    tableView.reloadData()  
}
```

## SEARCH BAR

---

### 3. Não funciona com maiúsculas/minúsculas diferentes

- ▶ Queremos padronizar a consulta
- ▶ Vamos colocar todos os textos em minúsculo

```
func searchBar(_ searchBar: UISearchBar, textDidChange searchText: String) {  
    if (searchText == "") {  
        arrayFiltered = arrayNome  
    } else {  
        arrayFiltered = arrayNome.filter { $0.lowercased().range(of: searchText.lowercased()) != nil }  
    }  
    tableView.reloadData()  
}
```



---

# CAMADA MODEL

---

# ANTES, UM REVIEW MVC

## CAMADAS

---

- ▶ Camadas são abstrações para facilitar o entendimento do código
  - ▶ Diminuem o tamanho das classes - não fica tudo na ViewController
  - ▶ Ajudam muito no refactoring
  - ▶ Ajudam quando outras pessoas trabalham no projeto
  - ▶ De uma forma BEM grosseira, são agrupamentos de arquivos de acordo com certa funcionalidade

## CAMADA VIEW

---

- ▶ Nós vimos muito a camada View
  - ▶ Ela é composta por tudo aquilo relativo ao que será visível na tela;
  - ▶ Arquivos comuns de fazerem parte são classes que herdam de:
    - ▶ UIViewController                      ▶ UIButton
    - ▶ UITableViewController           ▶ UINavigationController - ( customização de uma NavBar)
    - ▶ SegmentedControl           ▶ TabBarController - ( customização de uma TabBar)
  - ▶ Uma View se comunica com a outra seguindo o padrão **Delegate**
  - ▶ Se você tem que alterar algo visual, você irá encontrar na camada View

## CAMADA CONTROLLER

---

- ▶ Nós vimos um pouco da camada Controller:
  - ▶ Ela que executa os cálculos que são exibidos pelas Views
  - ▶ Provê a informação mais atual para ser enviada para a View e então exibida
  - ▶ Tem a responsabilidade de orquestrar a comunicação com o servidor e banco de dados local
  - ▶ Para se comunicar com a View, BD ou servidor temos que analisar o seguinte:
    - ▶ A busca dessa informação vai demorar (ex: servidor) - Vamos nos comunicar por closures
    - ▶ A busca dessa informação é imediata (ex: função calcular raiz quadrada) - vamos nos comunicar por uma função simples

---

**VOLTANDO...**



## CAMADA MODEL

---

- ▶ É como nossas informações estão estruturadas
- ▶ É um modelo das informações que é transformado em uma classe (Ex: classe usuario)
- ▶ A instância de uma Model é a representação mais atual dos seus dados.
- ▶ Para isso, são eles que são atualizados ao consultar informações no servidor ou no seu banco de dados
- ▶ Alterações nessas classes devem ser muito cautelosas, pois podem implicar em uma migração do banco de dados ou uma mudança no envio do servidor

---

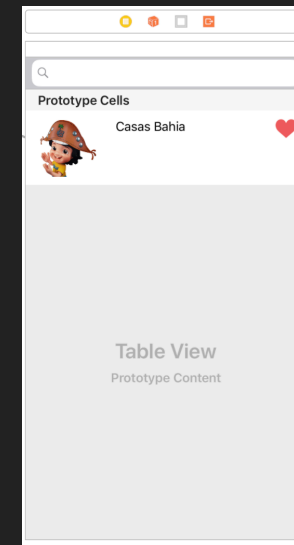
**NO INÍCIO...**

# TUDO NA VIEWCONTROLLER

```
class BuscaLojasTableViewController: UITableViewController {  
  
    let nomeLojas = ["Casas Bahia", "Ricardo Electro", "Juninho System", "Americanas"]  
    let logoLoja = [👤, 📺, 🛒, ❤️]
```

```
override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {  
    return nomeLojas.count  
}
```

```
if let cellLoja = cell as? LojaTableViewCell {  
    cellLoja.nomeLoja.text = nomeLojas[indexPath.row]
```



---

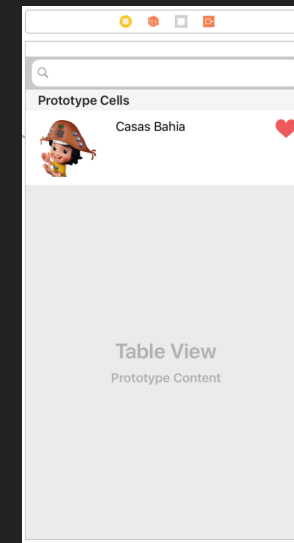
**MELHORAMOS...**

# VIEWCONTROLLER

```
class MinhalistaEstaticaTableViewController: UITableViewController, UISearchBarDelegate {  
    let controller = LojasController()
```

```
override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {  
    return controller.getQuantidadeLojas()  
}
```

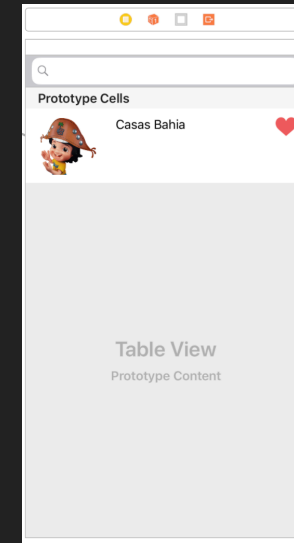
```
if let cellLojas = cell as? LojasTableViewCell {  
    cellLojas.nomeLoja.text = controller.nomeLoja(indexPath.row)
```



CAMADA MODEL

# VIEWCONTROLLER & CONTROLLER

```
class LojasController {  
  
    let nomeLojas = ["Casas Bahia", "Ricardo Electro", "Juninho System", "Americanas"]  
    let logoLoja = [🏠, 📺, 🛒, ❤️]  
  
    func getQuantidadeLojas() -> Int {  
        return nomeLojas.count  
    }  
  
    func nomeLoja(_ index: Int?) -> String {  
        guard let index = index, index < nomeLojas.count else {  
            return ""  
        }  
        return nomeLojas[index]  
    }  
}
```



## CAMADA MODEL

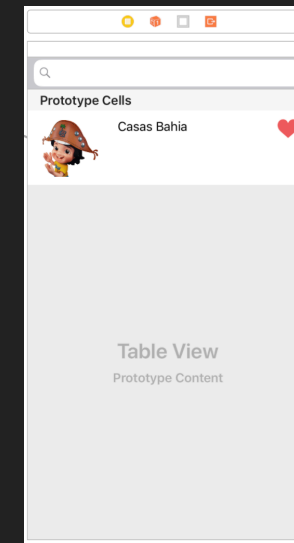
Na prática:

- ▶ Tínhamos dois arrays para controlar essa tela:

arrayNomes

arrayLogos

- ▶ Se fossemos colocar o favorito e as duas imagens seriam mais outros dois arrays.... Fica complicado.
- ▶ Vamos criar uma classe lojas para agrupar essas informações. Cada loja terá um nome, o nome da imagem, favorito e se tem computador ou jogos ou ambos.



---

**AGORA...**

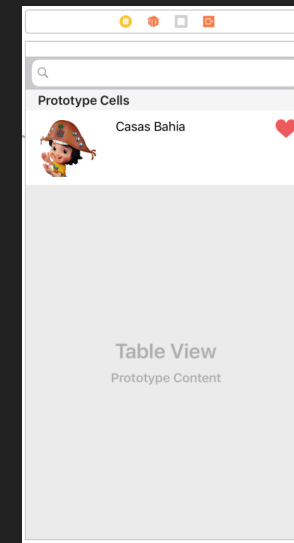


# VIEWCONTROLLER

```
class MinhalistaEstaticaTableViewController: UITableViewController, UISearchBarDelegate {  
    let controller = LojasController()
```

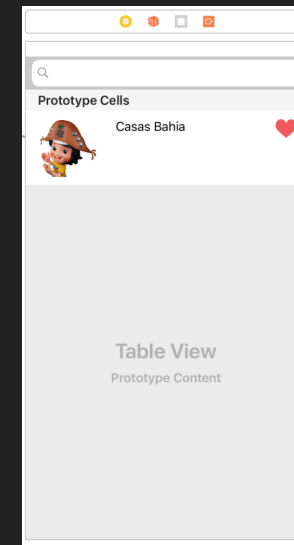
```
override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {  
    return controller.getQuantidadeLojas()  
}
```

```
if let cellLojas = cell as? LojasTableViewCell {  
    cellLojas.nomeLoja.text = controller.nomeLoja(indexPath.row)
```



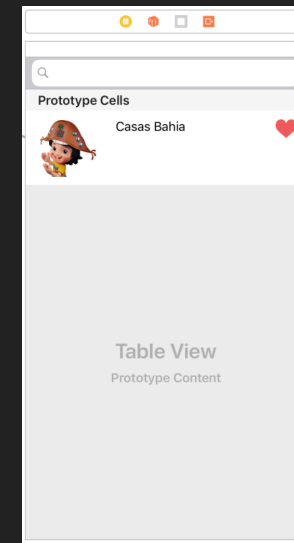
# VIEWCONTROLLER & CONTROLLER

```
class LojasController {  
  
    var arrayLojas: [Loja] = []  
  
    init() {  
        arrayLojas = buscarLojas()  
    }  
  
    func buscarLojas() -> [Loja] {  
        return [Loja(nome: "Casas Bahia", logo: "header-bahianinho"),  
                Loja(nome: "Ricardo Electro", logo: "header-ricardo"),  
                Loja(nome: "Juninho Systems", logo: "logo")]  
    }  
  
    func getQuantidadeLojas() -> Int {  
        return arrayLojas.count  
    }  
  
    func nomeLoja(_ index: Int?) -> String {  
        guard let index = index, index < arrayLojas.count else {  
            return ""  
        }  
        return arrayLojas[index].nome  
    }  
}
```



# VIEWCONTROLLER & CONTROLLER & MODEL

```
class Loja {  
    var logo: String  
    var nome: String  
  
    init(nome: String, logo: String) {  
        self.logo = logo  
        self.nome = nome  
    }  
}
```



## CAMADA MODEL

Podemos organizar nosso projeto em três pastas:

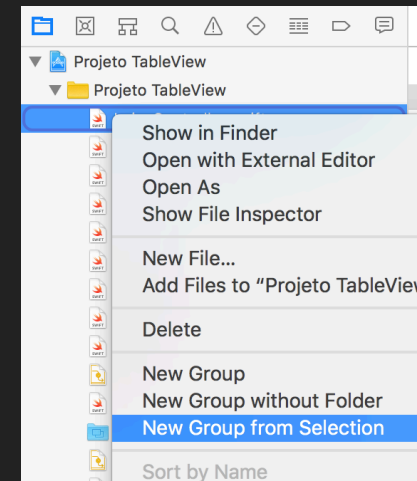
- ▶ Model
- ▶ View
- ▶ Controller

Para criar uma pasta, basta clicar com o botão direito em um arquivo e selecionar:

New Group

Se você quiser adicionar já um arquivo a pasta, selecione:

New Group From Selection



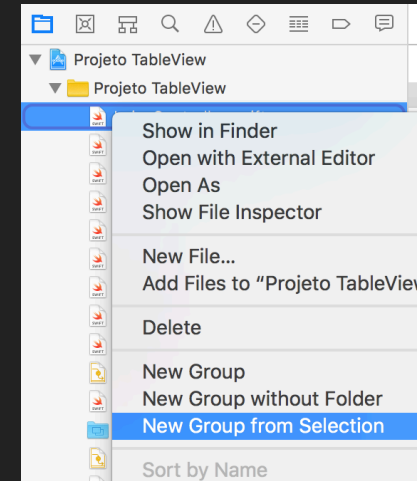
## CAMADA MODEL

Para mover um arquivo para outra pasta basta clicar e arrastar.

Dentro de cada uma das três pastas você pode criar novas pastas de acordo com a funcionalidade.

Outras pastas que podem existir são as pastas:

- ▶ Utils - Contém arquivos com constantes, Extensions
- ▶ Supporting Files - Contém arquivos de mídias, arquivos para fontes e outros arquivos úteis ao app
- ▶ Connection - Classes relacionados com a montagem da URL e conexão com o servidor - é discutível se não deve ser incorporado na Controller



# VOLTAMOS PARA O NOSSO PROJETO

