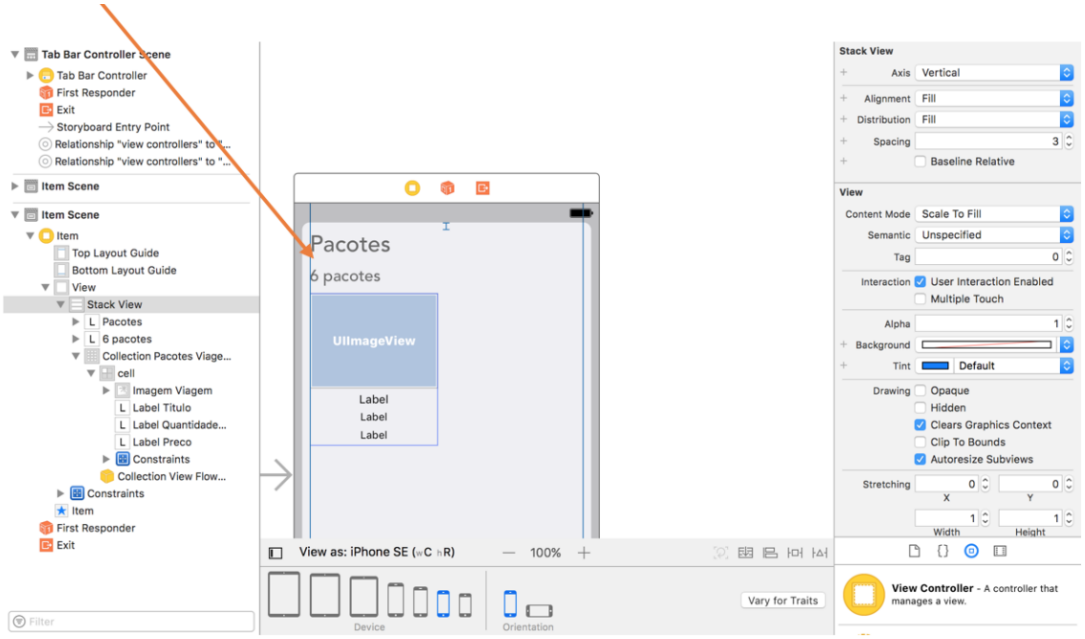
04

Filtrando as viagens utilizando NSPredicate

De volta ao nosso projeto, vamos voltar no ViewController de pacote de viagens. Quando estávamos implementando o layout dessa tela, colocamos um textfield para simular um elemento de busca. Porém, no storyboard já temos um elemento nativo para fazer o que precisamos.

Vamos abrir o storyboard e apagar o textfield que havíamos colocado:



Como o textfield estava dentro do stackview, quando o apagamos, os elementos se reposicionaram automaticamente e o storyboard não apontou nenhum erro.

1) Agora é sua vez: Coloque um 'Search Bar' no lugar do textfield.

O search bar é responsável por filtrar as viagens que o usuário digita. Assim como já implementamos vários métodos de delegate dos elementos (TableView, CollectionView...), o search bar também tem o seu protocolo que contém vários métodos que podemos utilizar.

Então vamos lá, vamos implementar o protocolo de `UISearchBarDelegate` no ViewController:

```
class PacotesViagensViewController: UIViewController, UICollectionViewDataSource, UICollectionV
```

Como já fizemos outras vezes, sempre que precisamos utilizar os métodos de delegate precisamos setar no viewcontroller:

```
override func viewDidLoad() {
        super.viewDidLoad()
        self.collectionPacotesViagens.dataSource = self
        self.collectionPacotesViagens.delegate = self
        self.searchViagens.delegate = self
}
```

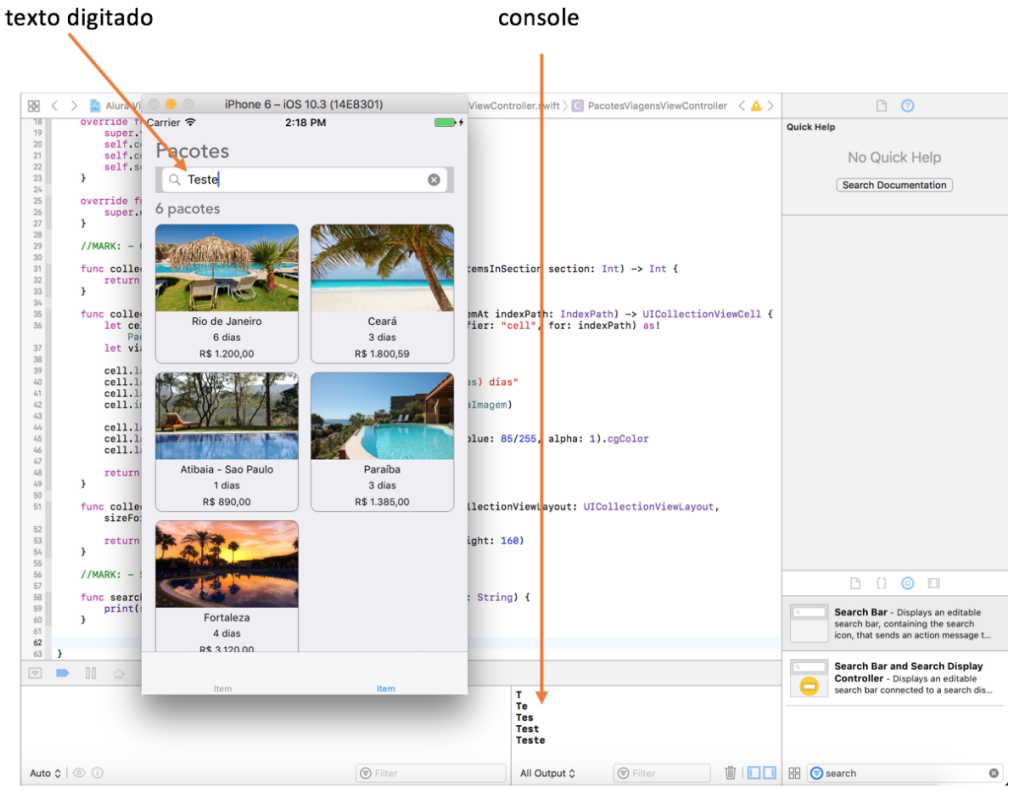
2) Agora é sua vez: Implemente o método: `textDidChange`

Toda vez que o usuário digitar algum caracter no search bar, esse método é disparado. Vamos ver se está funcionando. Para testar vamos imprimir no console os caracteres que digitarmos no simulador

Vamos colocar um 'print' dentro do método com o valor do 'searchBar':

```
func searchBar(_ searchBar: UISearchBar, textDidChange searchText: String) {
        print(searchBar.text!)
}
```

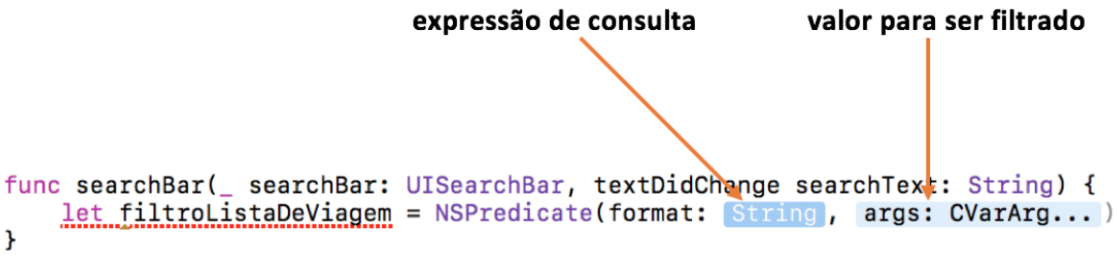
Agora vamos rodar o aplicativo no simulador e digitar alguma coisa no searchBar:



Como pode ver, agora conseguimos capturar o que o usuário está digitando. O próximo passo é filtrar a nossa lista de viagens com base no que foi digitado.

Existe uma classe do Swift que podemos utilizar para filtrar objetos em memória (como por exemplo, o array que estamos utilizando). Essa classe chama-se 'NSPredicate'

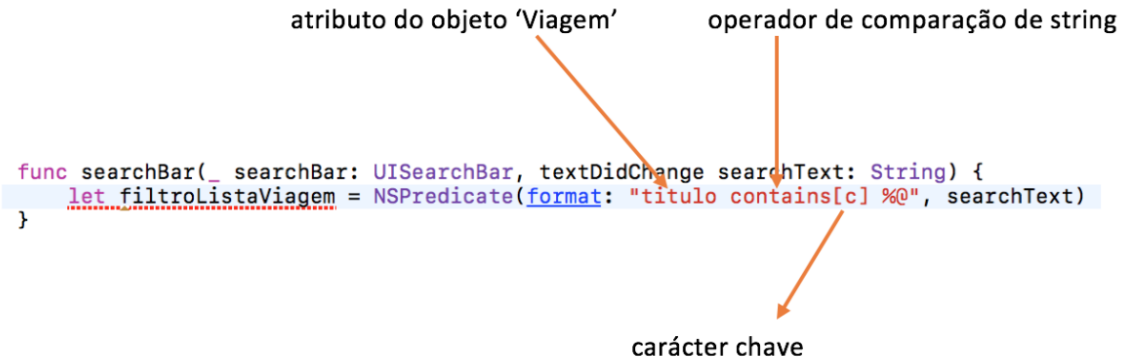
Então vamos implementá-la:



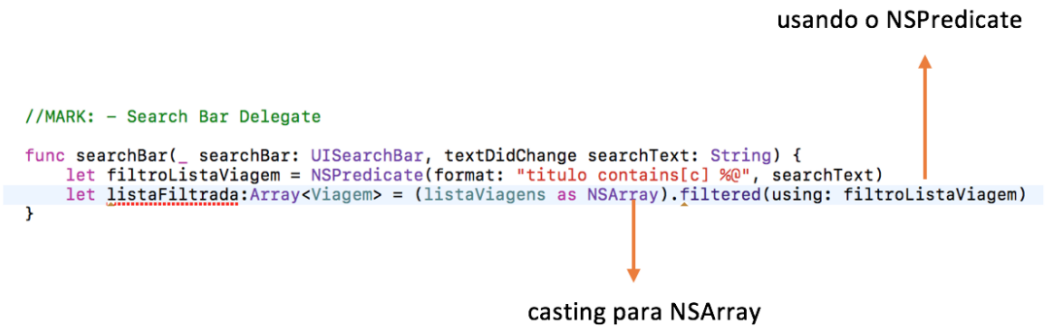
Quando estamos inicializando a classe do 'NSPredicate' temos vários métodos construtores. O que precisamos é um que passe uma 'expressão' para filtro e um valor para ser filtrado.

No nosso caso, temos uma lista com vários objetos 'Viagem' dentro. Precisamos escolher um atributo desse objeto que utilizaremos como parâmetro de busca. Nesse caso, o que faz mais sentido é o usuário pesquisar a viagem pelo título, não acha?

Então vamos lá:



Com o predicate criado, agora precisamos criar uma nova lista que guardará o resultado dos objetos filtrados na busca:



Como tipamos essa nova lista que criamos para 'Array', temos que fazer o casting novamente de 'NSArray' para 'Array'

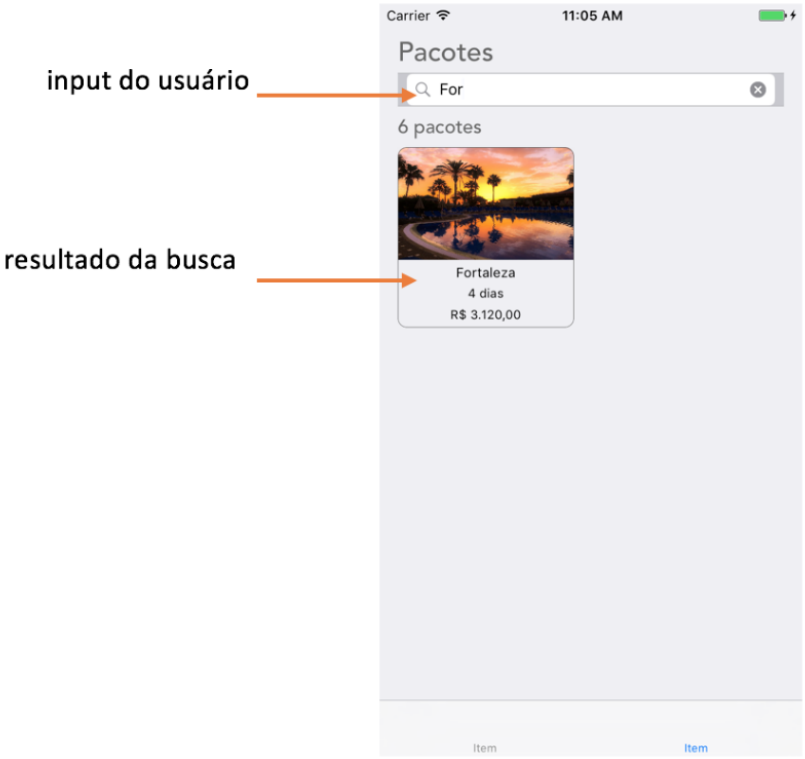
```
let listaFiltrada:Array<Viagem> = (listaComTodasAsViagens as NSArray).filtered(using: filtroLis
```

Para testar nossa implementação, vamos pegar o array que utilizamos no numberOfItens da CollectionView e igualar ao array que aplicamos o filtro e em seguida dar um reload na collection:

```
func searchBar(_ searchBar: UISearchBar, textDidChange searchText: String) {
        let filtroListaViagem = NSPredicate(format: "titulo contains %@", searchText)
        let listaFiltrada:Array<Viagem> = (listaViagens as NSArray).filtered(using: filtroLista
        listaViagens = listaFiltrada
```

```
            colecaoPacotesViagem.reloadData()
    }
```

Vamos rodar o app para testar:



Ótimo, o filtro já está funcionando.