DESENVOLVIMENTO PARA IOS – APPLE 01

IOS SDK 9

Professor: Pedro Henrique
prof.pedrohenrique.iossdk@gmail.com

AGENDA

- Formas de construir um novo objeto;
- Tipagem dinâmica;
- Introspecção e Selector;
- UITabBarController;
- UICollectionViewController;
- Introdução:
  - Reconhecimento de gestos;
  - Autolayout

OBJETIVOS DO DIA

- Dominar os meios de criar objetos;
- Compreender os mecanismos de tipagem dinâmica e introspecção;
- Aprender a usar o @selector;
- Aprender a usar os controles UITabBarController e UICollectionViewController;
- Avançar no entendimento dos protocolos;
- Iniciar os estudos sobre Autolayout.

CRIAÇÃO DE OBJETOS - CONSTRUTORES

```objc
NSArray *array = [[NSArray alloc] init];
```

```objc
NSMutableDictionary *dictionary = [[NSMutableDictionary alloc] initWithCapacity:10];
```

```objc
UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Mensagem"
                                                message:@"Declare variables, not war."
                                               delegate:nil
                                      cancelButtonTitle:@"OK"
                                      otherButtonTitles:nil];
```

Novidade do dia!

```objc
NSSet *set = [NSSet new];
```

=

```objc
NSSet *set = [[NSSet alloc] init];
```

CRIAÇÃO DE OBJETOS - CONSTRUTORES

```objc
NSMutableDictionary *dictionary = [[NSMutableDictionary alloc] initWithCapacity:10];
```

Designated Initializer

Designated Initializer

```objc
UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Mensagem"
                                                message:@"Declare variables, not war."
                                               delegate:nil
                                      cancelButtonTitle:@"OK"
                                      otherButtonTitles:nil];
```

CRIAÇÃO DE OBJETOS - CONSTRUTORES

```
NSSet *set = [NSSet new];
```

Somente usa o inicializador padrão. Dito isso, nenhum *designated initizalizer* poderá ser usado com esta sintaxe.

CRIAÇÃO DE OBJETOS – MÉTODOS DE CLASSE

```
NSString *minhaIdade = [NSString stringWithFormat:@"Minha idade é: %d", 18];

UIButton *botao = [UIButton buttonWithType:UIButtonTypeSystem];

NSArray *array = [NSArray arrayWithObjects:@"Carro", @"Bike", @"Moto", nil];
```

- Internamente, estes métodos usam um inicializador, seja o padrão ou um *designated* para construir e entregar uma nova instância do objeto;
- Esta construção existe para oferecer um atalho na hora de codificar. É aplicável a outras linguagens!

CRIAÇÃO DE OBJETOS - EXEMPLOS

```
12 @interface Exemplo ()
13 @property (strong, nonatomic) NSString *nome;
14 @property (strong, nonatomic) NSNumber *idade;
15 @end
16
17 @implementation Exemplo
18
19 - (instancetype) initWithNome: (NSString *) nome {
20     self = [super init];
21     if (self) {
22         [self setNome:nome];
23     }
24     return self;
25 }
26
27 + (instancetype) exemploComNome: (NSString *) nome {
28     return [[Exemplo alloc] initWithNome:nome];
29 }
30
31 + (instancetype) exemploComNome: (NSString *) nome idade: (NSNumber *) idade {
32     Exemplo *exemplo = [[Exemplo alloc] init];
33     [exemplo setNome:nome];
34     [exemplo setIdade:idade];
35     return exemplo;
36 }
37
38 @end
```

Símbolo "+" representa método de classe.

TIPAGEM DINÂMICA

- O Objective-C tem um tipo coringa muito importante, que se chama **id**;
  - O significado de **id** é:
    - Ponteiro para um objeto de tipo desconhecido ou não especificado.
- Em tempo de execução, TODOS os objetos são tratados como **id**;
- Requer muito cuidado ao usar!

TIPAGEM DINÂMICA

```
38 - (void) cuidadoComTipagemDinamica {
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57 }
```

TIPAGEM DINÂMICA

```
20 @interface Veiculo : NSObject
21 - (void) mover;
22 @end
23
24 @interface Tanque : Veiculo
25 - (void) atirar;
26 @end
27
28
29 @implementation Exemplo
30 - (void) exemplo {
31
32     Tanque *tanque = [[Tanque al      init];
33     [tanque mover];
34     [tanque atirar];
35
36
37     Veiculo *v = tanque;
38     [v atirar];
39 }
40
41
42 @end
```

Alerta de compilação!

- Esse exemplo específico funcionaria, porque **v** é um Tanque;
- Mas o compilador não sabe disso.

INTROSPECÇÃO E SELECTOR

- Todos os objetos filhos de NSObject sabem fazer introspecção:
- **isKindOfClass:**
  - Retorna **YES**, se o objeto for do mesmo tipo do parâmetro (incluindo a árvore de herança)
- **isMemberOfClass:**
  - Idem anterior, porém ignora a herança
- **respondsToSelector:**
  - Retorna **YES** se o objeto for capaz de responder a uma dada mensagem (método)

INTROSPECÇÃO E SELECTOR

```
NSString *s = @"s";

[s isKindOfClass: [NSObject class]]; //YES

[s isMemberOfClass: [NSNumber class]]; //NO

[s respondsToSelector: @selector(rangeOfString:)]; //YES
```

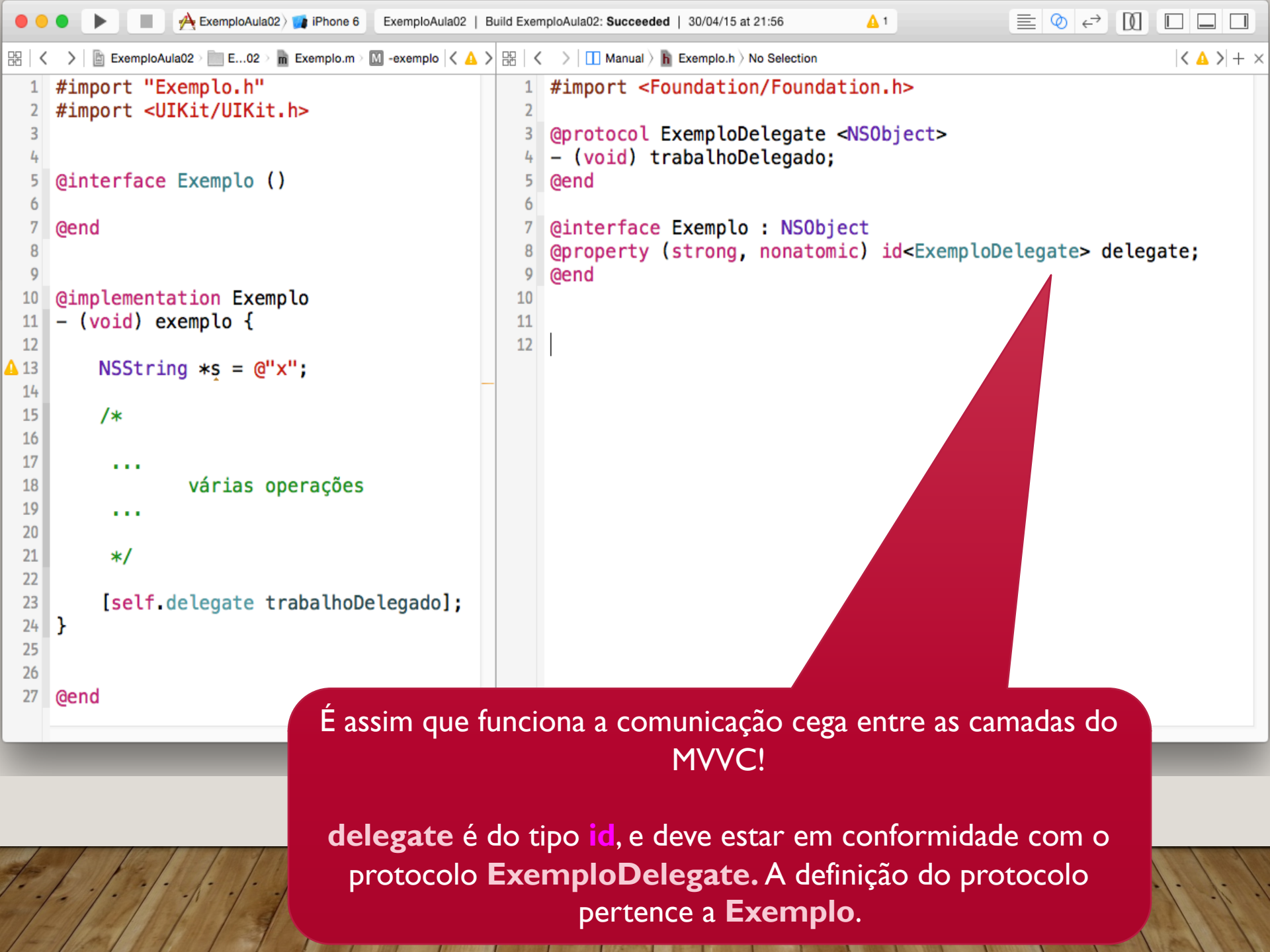
INTROSPECÇÃO E SELECTOR

Sintaxe para armazenar um **@selector** em uma variável

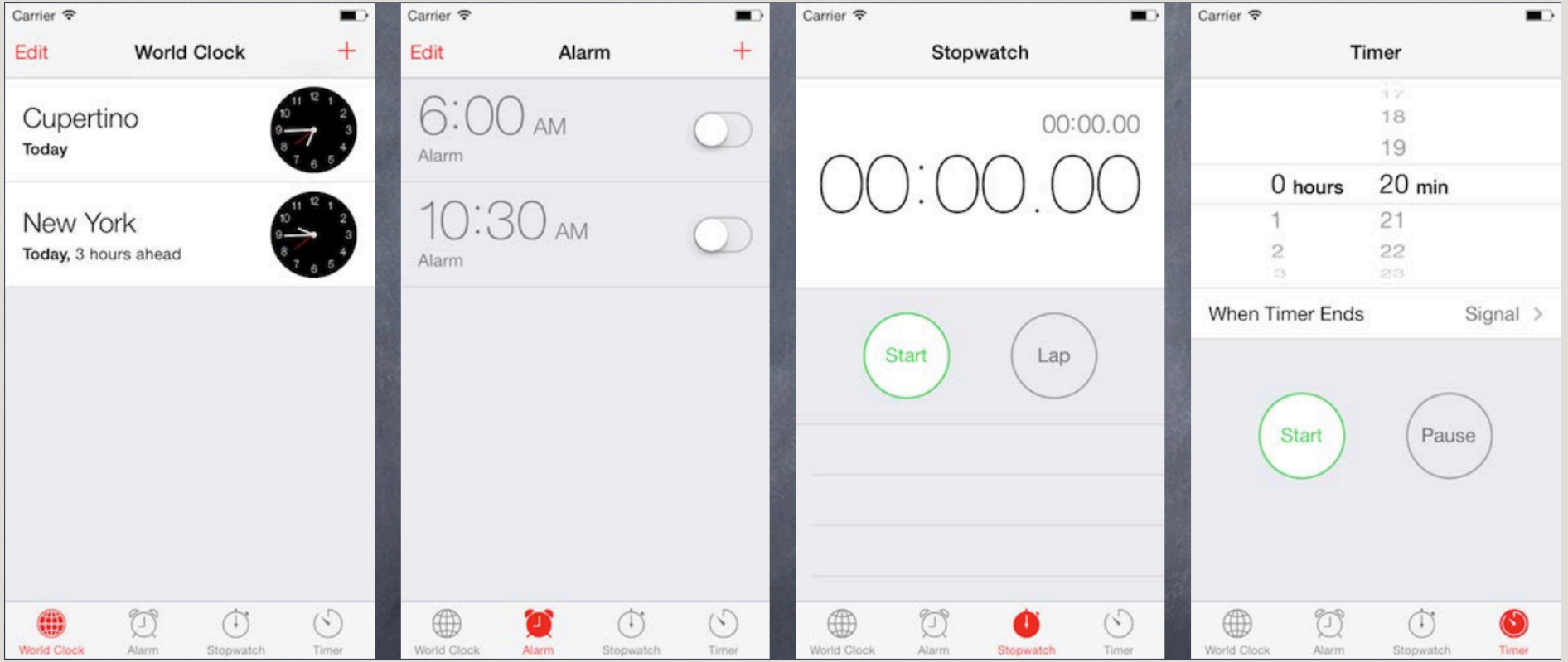
```
NSString *s = @"x";
NSArray *numeros = @[@1, @2, @3, @4];

SEL umSelector = @selector(description);
SEL selectorComArgumento = @selector(stringByAppendingString:);
SEL maisDeUmArgumento = @selector(arrayWithObjects:count:);

[s respondsToSelector:umSelector]; // retorna BOOL

[numeros makeObjectsPerformSelector:umSelector]; //retorna void

[s performSelector:umSelector]; //retorna id
NSString *s1 = [s performSelector:selectorComArgumento withObject:@"1"]; //retorna id
//s1 contém a string "x1"
```

PROTOCOLOS

- Juntando os conceitos de tipagem dinâmica e introspecção, o conceito de protocolo ganha um novo significado!


ExemploAula02 › iPhone 6
ExemploAula02 | Build ExemploAula02: Succeeded | 30/04/15 at 21:56
ExemploAula02 › E...02 › Exemplo.m › -exemplo
Manual › Exemplo.h › No Selection
#import "Exemplo.h"
#import <UIKit/UIKit.h>
@interface Exemplo ()
@end
@implementation Exemplo
- (void) exemplo {
NSString *s = @"x";
/*
...
várias operações
...
*/
[self.delegate trabalhoDelegado];
}
@end
#import <Foundation/Foundation.h>
@protocol ExemploDelegate <NSObject>
- (void) trabalhoDelegado;
@end
@interface Exemplo : NSObject
@property (strong, nonatomic) id<ExemploDelegate> delegate;
@end
É assim que funciona a comunicação cega entre as camadas do MVVC!
delegate é do tipo id, e deve estar em conformidade com o protocolo ExemploDelegate. A definição do protocolo pertence a Exemplo.

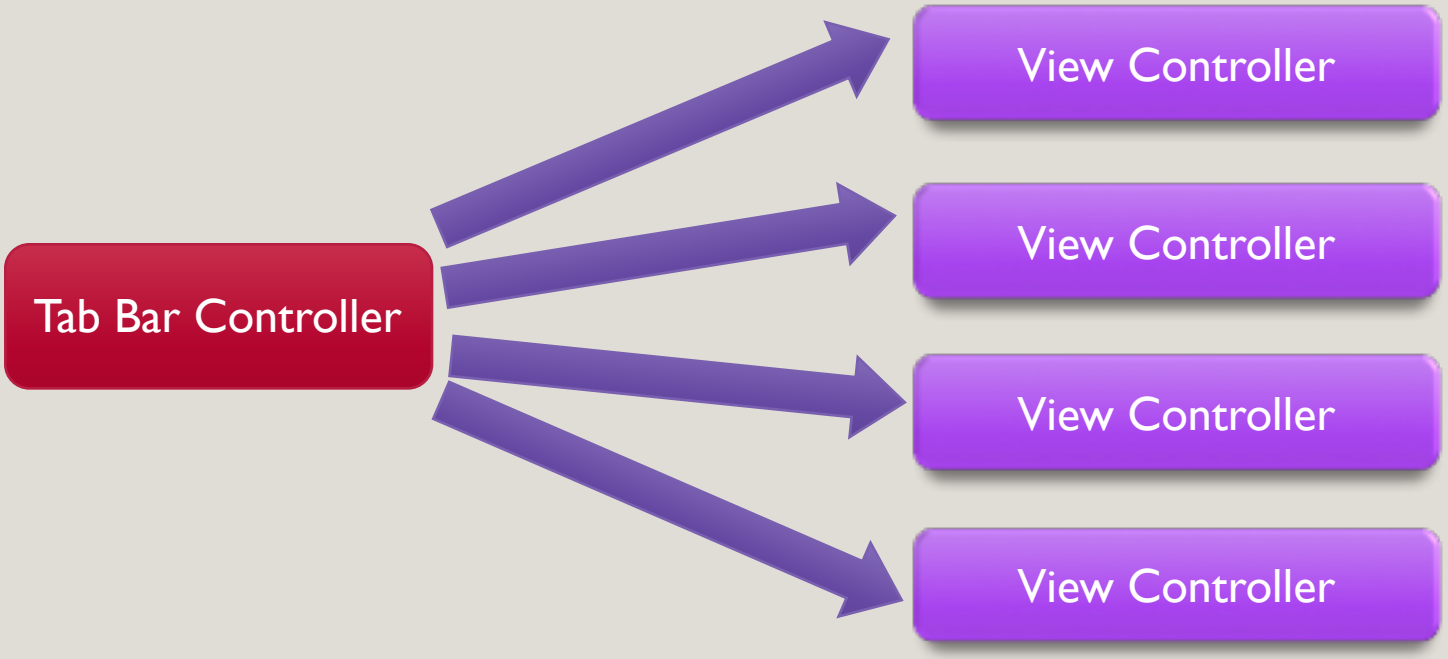
UITABBARCONTROLLER



UITABBARCONTROLLER

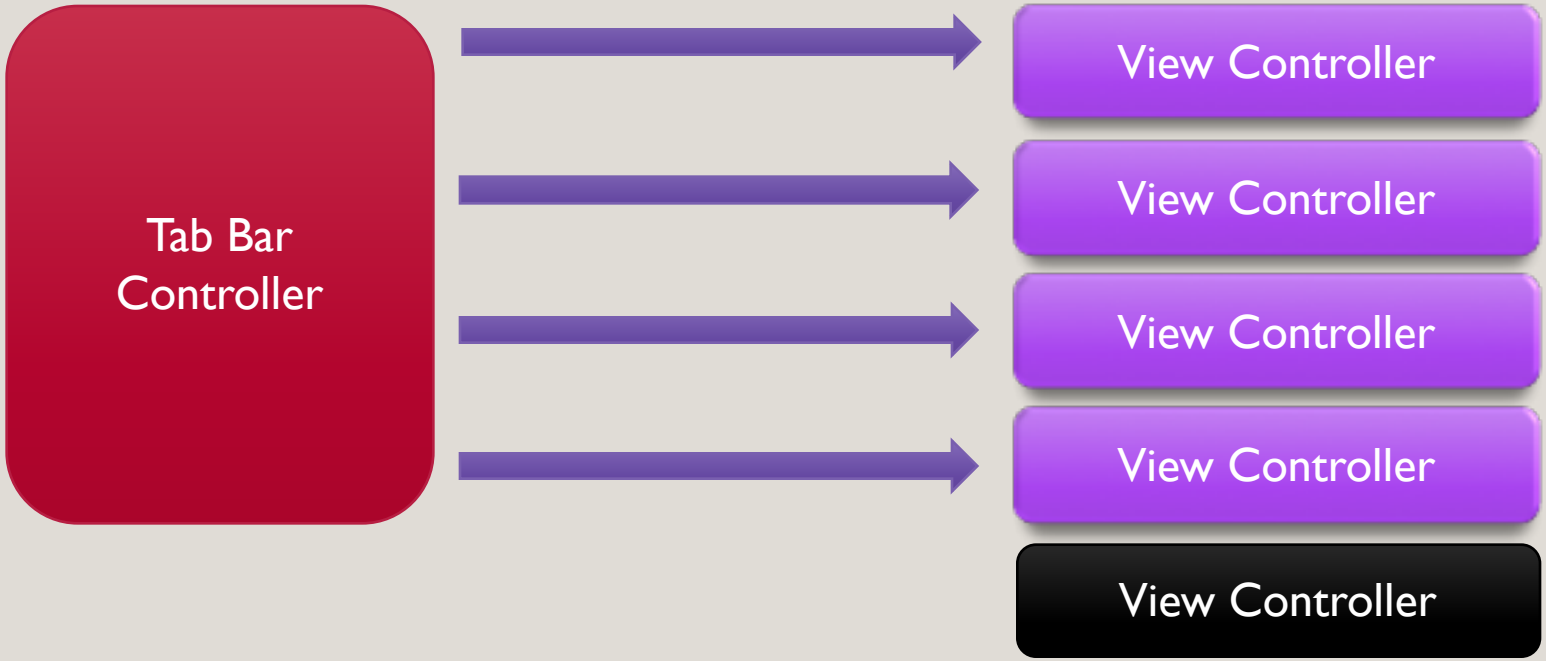
- Um dos principais componentes do iOS;
- Controla abas, onde cada aba é um UIViewController;
- Exibe o número de abas que o dispositivo é capaz de exibir sem comprometer o design;
- Cuidado!
  - O UITabBarController não pode estar na hierarquia de ViewControllers do UINavigationController!
  - O oposto é o correto, portanto.

UITABBARCONTROLLER

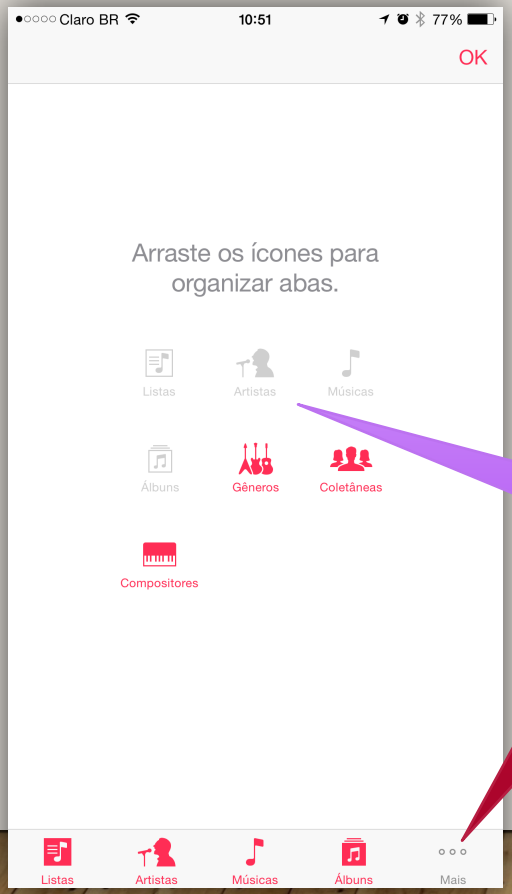


UITABBARCONTROLLER

- E se eu precisar de mais abas?



UITABBARCONTROLLER



Quando há mais do que 4 ViewControllers associados, o botão "Mais" aparece automaticamente.

Esta interface, que permite ao usuário reordenar as abas, já vem pronta e aparece automaticamente quando há mais de 4 abas!

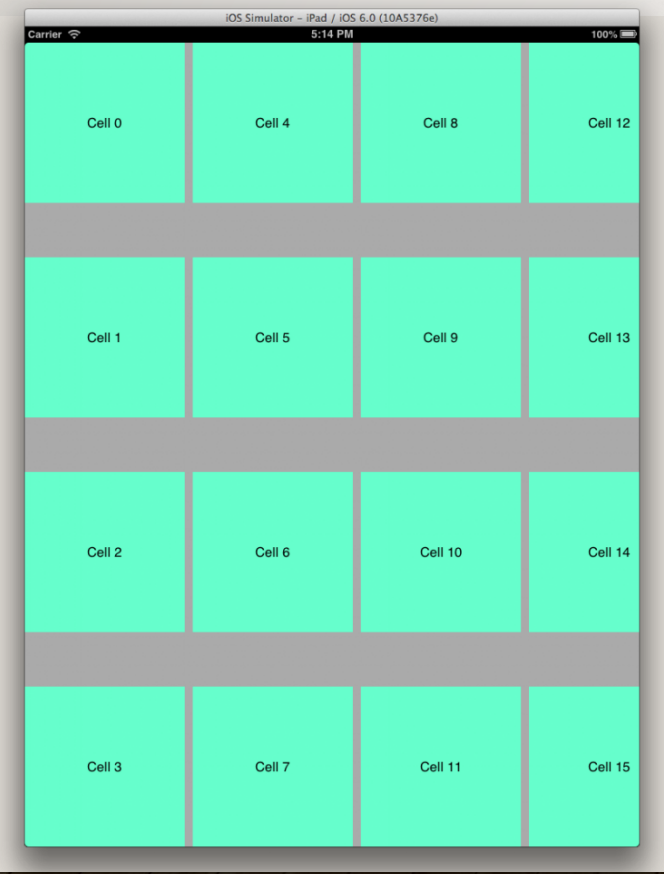
UICOLLECTIONVIEWCONTROLLER

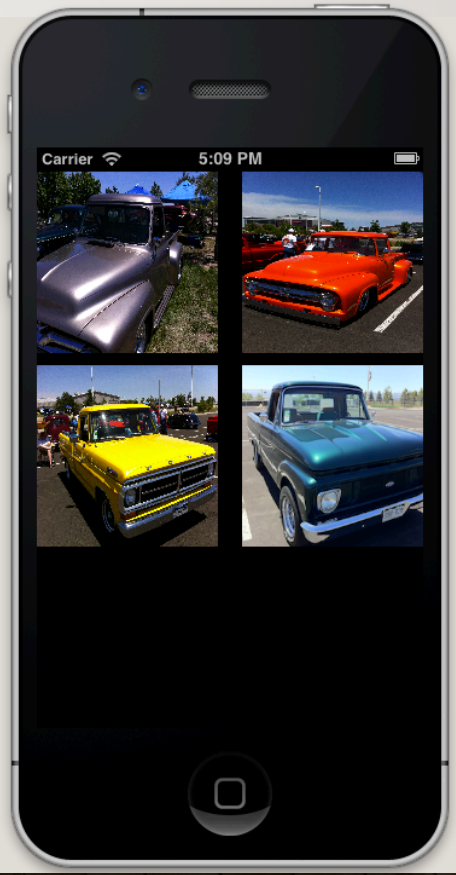
- Irmão do UITableViewController;
- Serve para exibir um *array* de views em um formato que não seja de lista;
- Um dos usos mais comuns é a construção de galerias de fotos;
- A organização mais usada é o formato de *grid*.

UICOLLECTIONVIEWCONTROLLER

- UICollectionViewCell;
  - E subclasses customizadas;
- UICollectionViewDataSource;
- UICollectionViewDelegate;
- UICollectionViewLayout;
  - UICollectionViewFlowLayout

UICOLLECTIONVIEWCONTROLLER





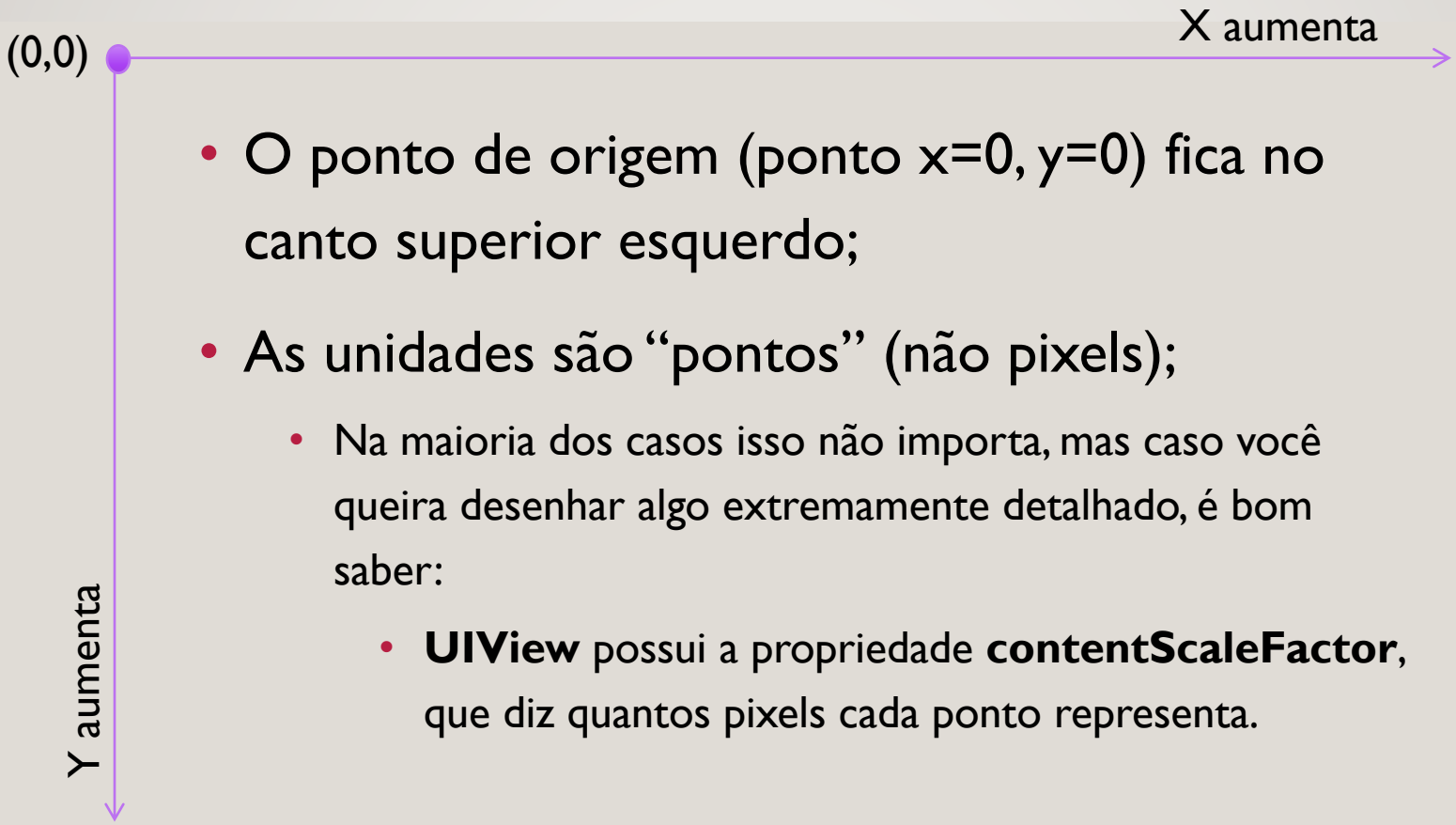
RECONHECIMENTO DE GESTOS

- Tap;
  - Toque (com um ou mais dedos)
- Swipe;
- Pan;
  - Arrastar
- Pinch;
- Rotation;
- **http://www.appcoda.com/ios-gesture-recognizers/**

INTRODUÇÃO AO AUTOLAYOUT

- Sistema de coordenadas;
- Sistema de unidades;
- Structs envolvidas;
  - Parte do framework CoreGraphics – inteiramente em linguagem C;

INTRODUÇÃO AO AUTOLAYOUT

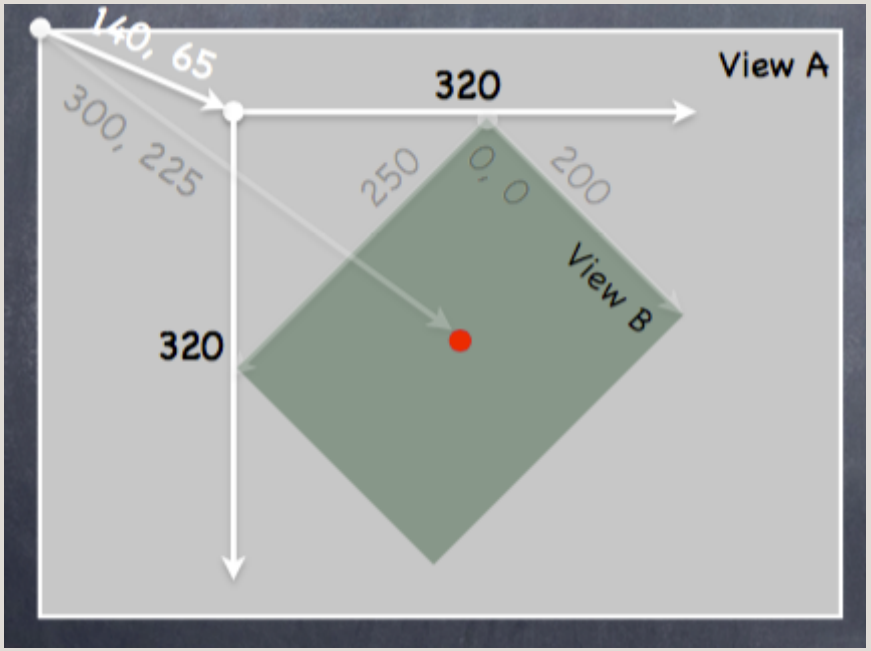


- O ponto de origem (ponto x=0, y=0) fica no canto superior esquerdo;
- As unidades são "pontos" (não pixels);
  - Na maioria dos casos isso não importa, mas caso você queira desenhar algo extremamente detalhado, é bom saber:
    - **UIView** possui a propriedade **contentScaleFactor**, que diz quantos pixels cada ponto representa.

INTRODUÇÃO AO AUTOLAYOUT

- Mais propriedades interessantes:
- **@property CGRect** bounds;
  - O espaço interno e tamaho da sua View. É usada para a implementação interna da View.
- **@property CGPoint** center;
  - O centro da sua View no sistema de coordenadas da SuperView.
- **@property CGRect** frame;
  - Um retângulo na SuperView que contém inteiramente a sua view (bounds.size)

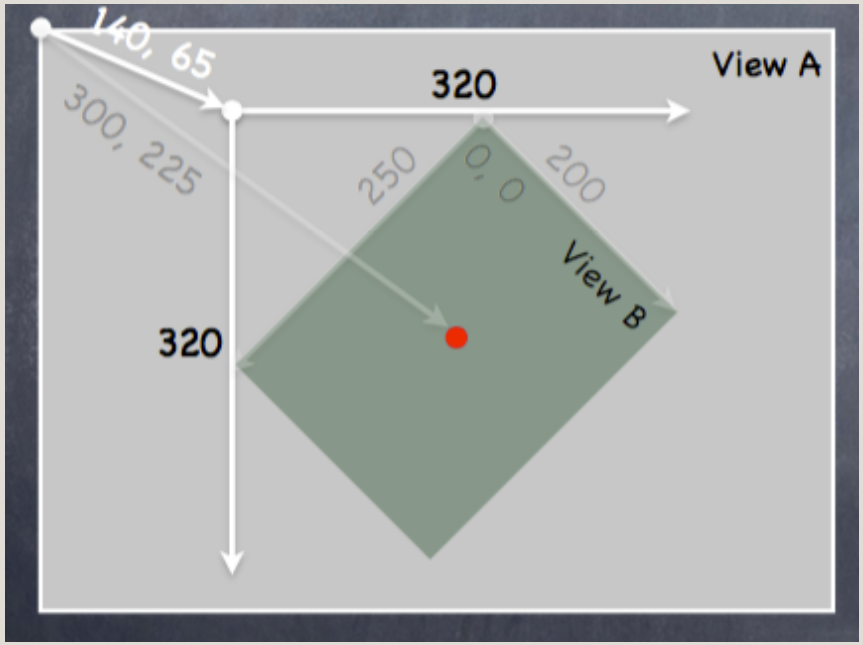
INTRODUÇÃO AO AUTOLAYOUT



- Views podem passar por transformações (escala, rotação e translação), portanto cuidado:
  - **bounds** de B = ((0,0), (250,250))
  - **frame** de B = ((140,65), (320,320))
  - **center** de B = (300,225)

INTRODUÇÃO AO AUTOLAYOUT

- Views raramente são rotacionadas, mas tome cuidado para não fazer confuzão com **frame** e **center**!



INTRODUÇÃO AO AUTOLAYOUT

- Por padrão, quando o tamanho de uma view (**bounds**) muda, não ocorre redesenho;
  - Em vez disso, a *view* é esticada, encolhida ou ainda reposicionada.
- Eventualmente pode não ser esse o comportamento que você deseja...
  - **@property UIViewContentMode** contentMode;

INTRODUÇÃO AO AUTOLAYOUT

- **UIViewContentMode**
  {Left,Right,Top,Right,BottomLeft,BottomRight,TopLeft,TopRight}
  - Move a view para a respectiva localização
- **UIViewContentModeScale**
  {ToFill,AspectFill,AspectFit}
  - Escala a view (estica ou encolhe)
- **UIViewContentModeRedraw**
  - Redesenha a view.
- Qual o padrão? **UIViewContentModeScaleToFill**

INTRODUÇÃO AO AUTOLAYOUT

- O que vem por aí?
- No futuro, vamos aprender a:
  - Lidar com frames usando regras em vez de números com uma API muito poderosa (NSLayoutConstraint)!
  - Mas, desde o Xcode 5, podemos fazer a maior parte do trabalho graficamente no InterfaceBuilder.

THE TWO STATES OF EVERY PROGRAMMER

I AM A GOD.

I HAVE NO IDEA WHAT I'M DOING.

HORA DE BRINCAR

- NSRange
- NSUserDefaults
- Literais
- Arquivos PLIST
- UITabBarController
- UICollectionViewController;
- Reconhecimento de Gestos
- Desafio!

DESAFIO

- Carregar um NSArray de strings a partir de um arquivo plist;
- Mostrar todos os elementos em um UITableView;
- Ao clicar em uma célula, navegar para uma outra tela, onde o "detalhe" daquela célula é apresentado.
- Navegar para uma terceira tela através do reconhecimento de um gesto.