IBDesignables & *IBInspectables*

Kaê Angeli Coutinho

O que são *IBDesignable* e *IBInspectable*?

- Duas novas palavras reservadas adicionadas na sexta versão do IDE *Xcode*
- Exibição de classes de componentes gráficos personalizados diretamente pelo *Interface Builder*
- *IBDesignable* informa ao *Interface Builder* que um componente gráfico personalizado irá ser pré-renderizado pelo mesmo
- *IBInspectable* informa ao *Interface Builder* que uma propriedade ou atributo de um componente gráfico personalizado irá ser manuseável pelo mesmo

Porque utiliza-las?

- Melhorias em reutilização, testes e compartilhamento de componentes gráficos
- Otimização de tempo de desenvolvimento de componentes gráficos
- Visualização em tempo real no *Interface Builder*

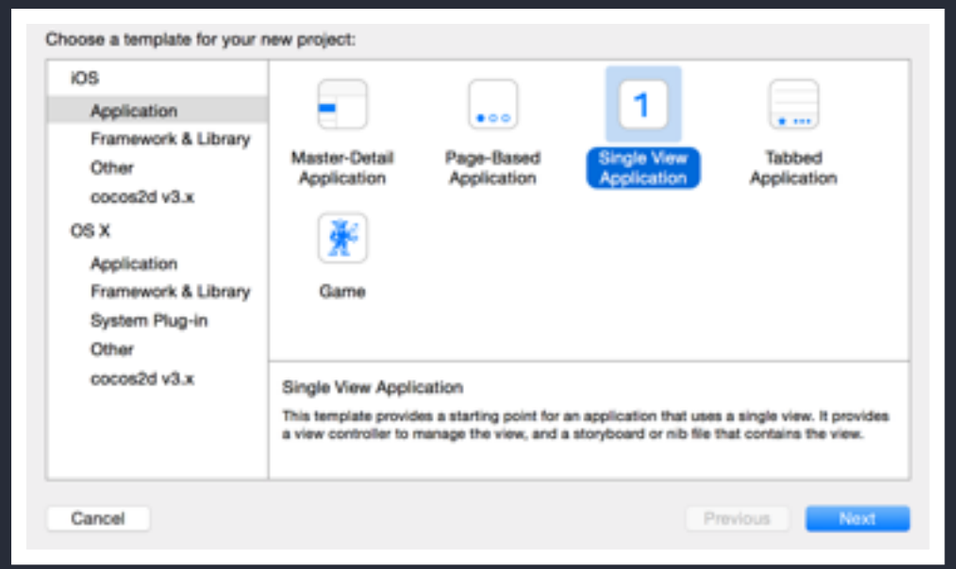
Como utiliza-las?

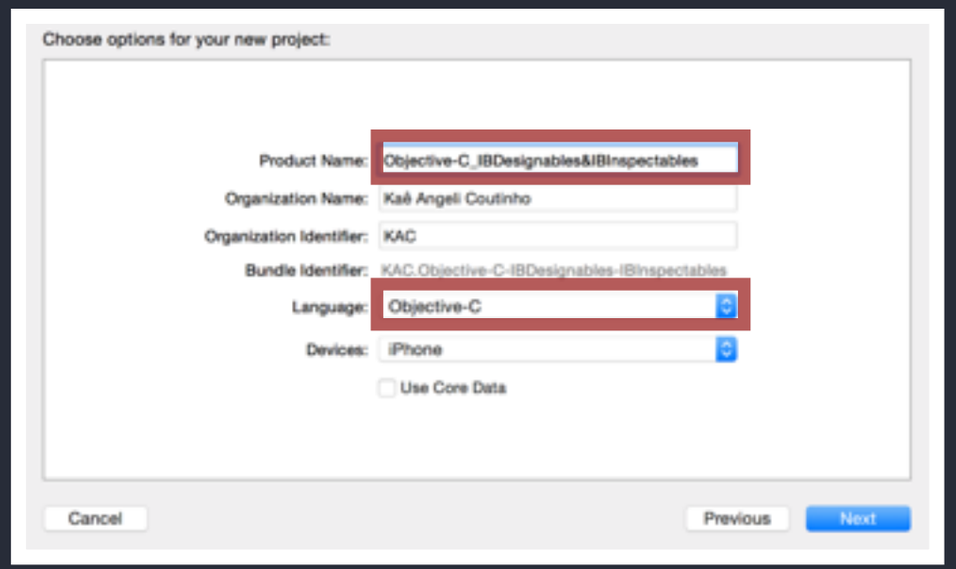
- É necessário a criação de um novo componente gráfico personalizado
- Declaração da palavra chave *IBDesignable* na criação da classe
- Declaração da palavra chave *IBInspectable* na criação de cada propriedade ou atributo

Mãos à obra

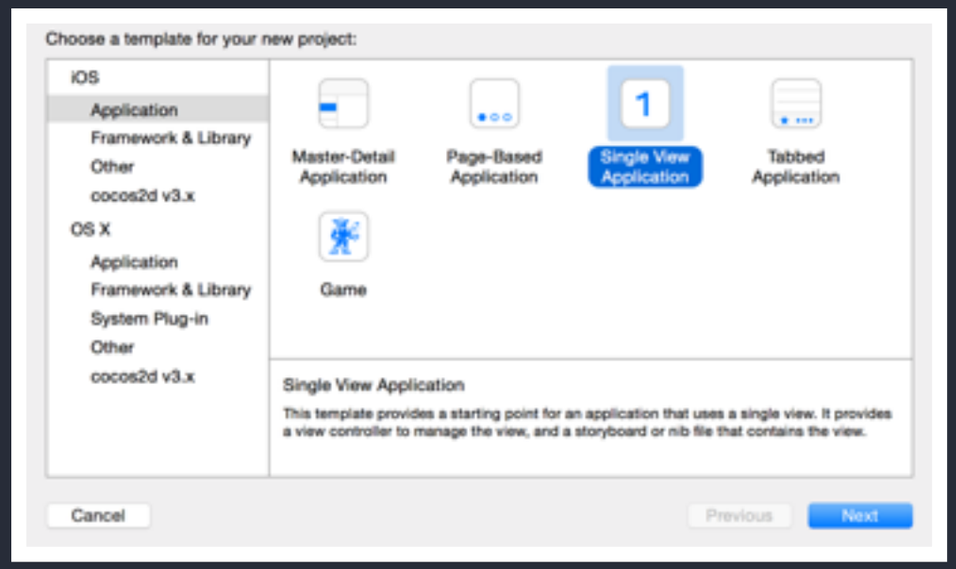
- Precisamos enfatizar que existe diferença de sintaxe para as palavras reservadas *IBDesignable* e *IBInspectable* entre as linguagens *Objective-C* e *Swift*
- *Objective-C (constantes)*
    - *IBDesignable: IB_DESIGNABLE*
    - *IBInspectable: IBInspectable*
- *Swift (anotações)*
    - *IBDesignable: @IBDesignable*
    - *IBInspectable: @IBInspectable*
- Como existe diferença enorme de sintaxe entre as duas linguagens de programação para a plataforma *iOS*, iremos conduzir esse tutorial com dois projetos, cada um direcionado para uma linguagem

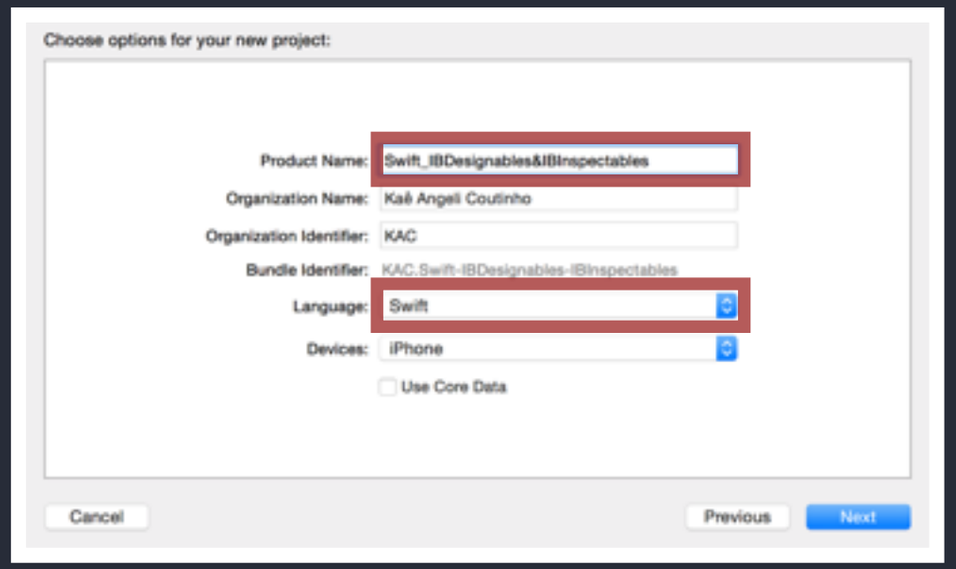
- Vamos começar dois novos projetos no Xcode
- Objective-C





- Swift

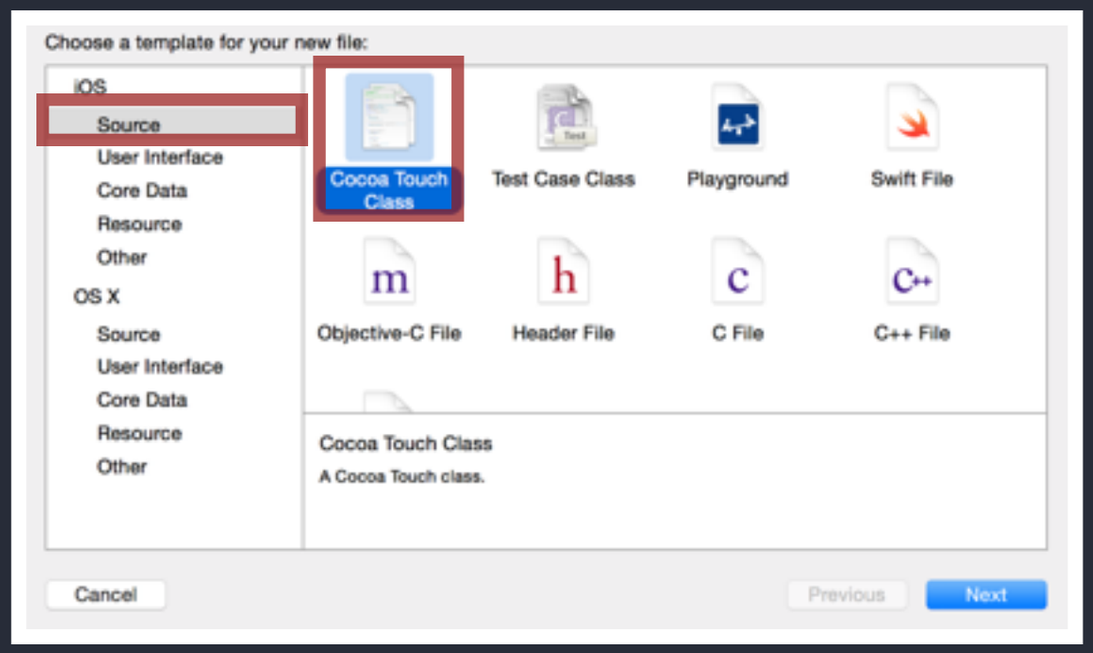


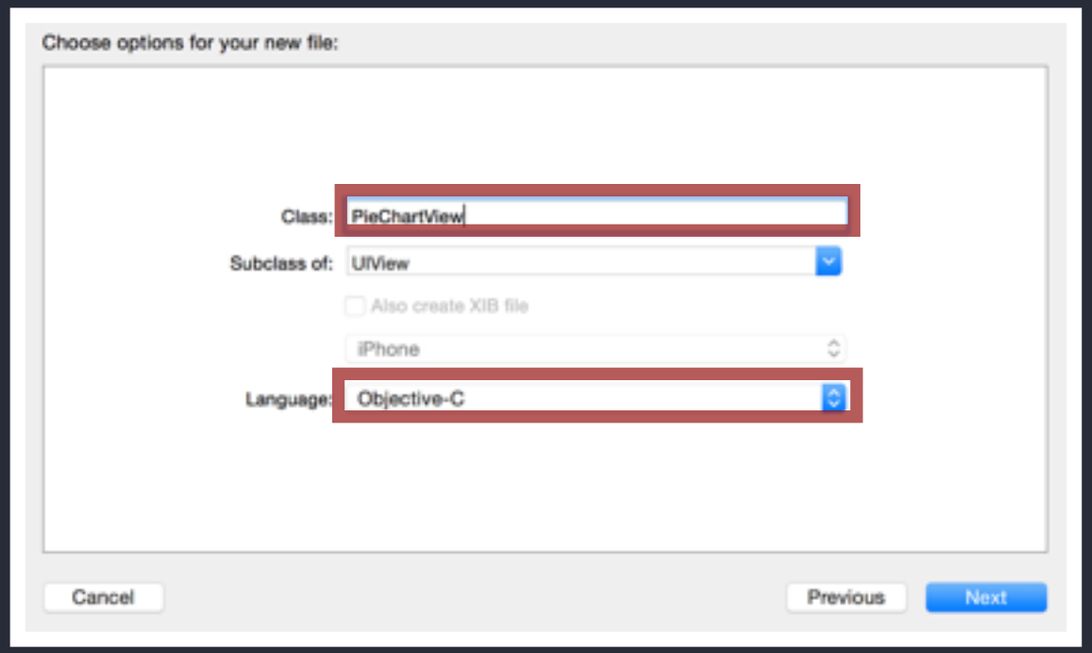


- Vamos primeiramente desenvolver o componente gráfico personalizado, para este tutorial iremos criar um gráfico circular

- *Objective-C*

  - Começaremos com a criação da nova classe PieChartView





- Começaremos com a declaração da classe (PieChartView.h)

```
#import <UIKit/UIKit.h>

IB_DESIGNABLE
@interface PieChartView : UIView

// Properties

@property (strong,nonatomic) CAShapeLayer * pieBackgroundLayer;
@property (strong,nonatomic) CAShapeLayer * pieLayer;
@property (assign,nonatomic) double pieStrokeWidth;
@property (assign,nonatomic) IBInspectable double piePercentage;
@property (strong,nonatomic) IBInspectable UIColor * pieBackgroundColor;
@property (strong,nonatomic) IBInspectable UIColor * pieColor;

// Methods

-(void)updateLayerProperties;
-(void)updatePiePercentage:(double)newPiePercentage;

@end
```

- Prosseguimos com a implementação da classe (PieChartView.m)
- É muito importante enfatizar que dois construtores irão ser criados e ambos são necessários, pois, cada um deles desemprenha uma função diferente na instanciação da classe
    - *-(void)initWithCoder:(NSCoder *)aDecoder* é invocado na execução do binário gerado quando a interface gráfica utilizada do mesmo fora storyboard
    - *-(void)initWithFrame:(CGRect)frame* é invocado na pré-renderização feita pelo Interface Builder, portanto, nenhum componente gráfico personalizado utilizando *IBDesignable* irá funcionar se este construtor não for desenvolvido

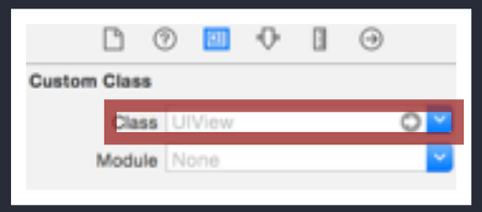
```
#pragma mark - Constructors

-(id)initWithCoder:(NSCoder *)aDecoder
{
    self = [super initWithCoder:aDecoder];
    if(self)
    {
        [self setPieBackgroundLayer:[CAShapeLayer new]];
        [self setPieLayer:[CAShapeLayer new]];
        [self setPieStrokeWidth:0.0];
        [self setPiePercentage:0.0];
        [self setPieBackgroundColor:[UIColor colorWithWhite:0.8 alpha:1.0]];
        [self setPieColor:[UIColor orangeColor]];
    }
    return self;
}

-(id)initWithFrame:(CGRect)frame
{
    self = [super initWithFrame:frame];
    if(self)
    {
        [self setPieBackgroundLayer:[CAShapeLayer new]];
        [self setPieLayer:[CAShapeLayer new]];
        [self setPieStrokeWidth:0.0];
        [self setPiePercentage:0.0];
        [self setPieBackgroundColor:[UIColor colorWithWhite:0.8 alpha:1.0]];
        [self setPieColor:[UIColor orangeColor]];
    }
    return self;
}
```

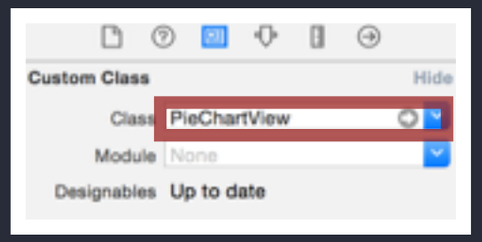
```
#pragma mark - Methods

-(void)layoutSubviews
{
    [super layoutSubviews];
    [self setPieStrokeWidth:[self frame].size.width / 2.0];
    if([self pieBackgroundLayer] != nil)
    {
        [self setPieBackgroundLayer:[CAShapeLayer new]];
        [[self layer] addSublayer:[self pieBackgroundLayer]];
        CGRect circleRect = CGRectInset([self bounds],
                                        ([self pieStrokeWidth] / 2.0),
                                        ([self pieStrokeWidth] / 2.0));
        UIBezierPath * circlePath = [UIBezierPath bezierPathWithOvalInRect:circleRect];
        [[self pieBackgroundLayer] setPath:[circlePath CGPath]];
        [[self pieBackgroundLayer] setFillColor:nil];
        [[self pieBackgroundLayer] setLineWidth:[self pieStrokeWidth]];
        [[self pieBackgroundLayer] setStrokeColor:[[self pieBackgroundColor] CGColor]];
    }
    [[self pieBackgroundLayer] setFrame:[[self layer] bounds]];
    if([self pieLayer] != nil)
    {
        [self setPieLayer:[CAShapeLayer new]];
        [[self layer] addSublayer:[self pieLayer]];
        CGRect circleRect = CGRectInset([self bounds],
                                        ([self pieStrokeWidth] / 2.0),
                                        ([self pieStrokeWidth] / 2.0));
        UIBezierPath * circlePath = [UIBezierPath bezierPathWithOvalInRect:circleRect];
        [[self pieLayer] setPath:[circlePath CGPath]];
        [[self pieLayer] setFillColor:nil];
        [[self pieLayer] setLineWidth:[self pieStrokeWidth]];
        [[self pieLayer] setStrokeColor:[[self pieColor] CGColor]];

        [[self pieLayer] setAnchorPoint:CGPointMake(0.5,0.5)];
        [[self pieLayer] setTransform:CATransform3DRotate([[self pieLayer] transform],(-M_PI / 2.0),0.0,0.0,1.0)];
    }
    [[self pieLayer] setFrame:[[self layer] bounds]];
    [self updateLayerProperties];
}
```

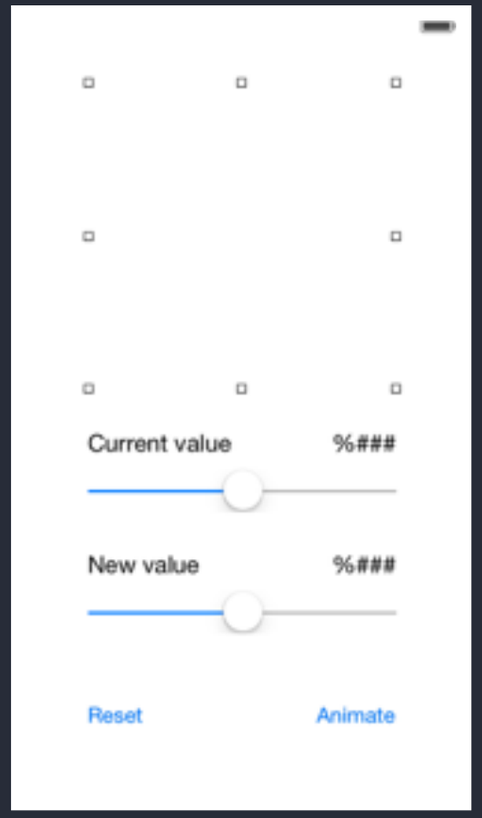
```
-(void)updateLayerProperties
{
    if([self pieLayer] != nil)
    {
        [[self pieLayer] setStrokeEnd:([self piePercentage] / 100.0)];
    }
}
```

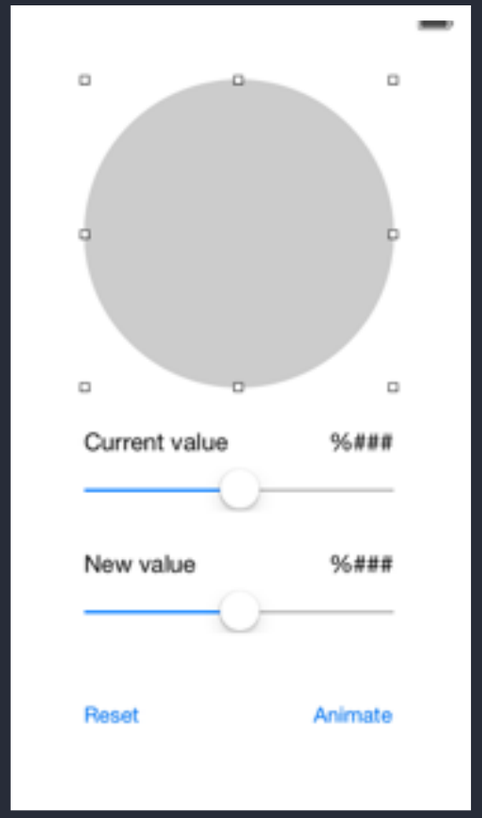
```
-(void)updatePiePercentage:(double)newPiePercentage
{
    if([self pieLayer] != nil)
    {
        [CATransaction begin];
        CABasicAnimation * animation = [CABasicAnimation animationWithKeyPath:@"strokeEnd"];
        [animation setDuration:((newPiePercentage / 100.0) - ([self piePercentage] / 100.0)) * 3.0];
        [animation setFromValue:[NSNumber numberWithDouble:[self piePercentage] / 100.0]];
        [animation setToValue:[NSNumber numberWithDouble:(newPiePercentage / 100.0)]];
        [animation setTimingFunction:[CAMediaTimingFunction functionWithName:kCAMediaTimingFunctionEaseInEaseOut]];
        [CATransaction setCompletionBlock:^
        {
            [CATransaction begin];
            [CATransaction setValue:kCFBooleanTrue forKey:kCATransactionDisableActions];
            [[self pieLayer] setStrokeEnd:(newPiePercentage / 100.0)];
            [CATransaction commit];
        }];
        [[self pieLayer] addAnimation:animation forKey:@"animateStrokeEnd"];
        [CATransaction commit];
    }
}
```

- Iremos agora desenvolver nossa interface gráfica (storyboard)
- Reparem que nossa UIView não fora ainda renderizada pelo *Interface Builder*
- É necessário atrelar a classe do componente gráfico customizado recém-criado

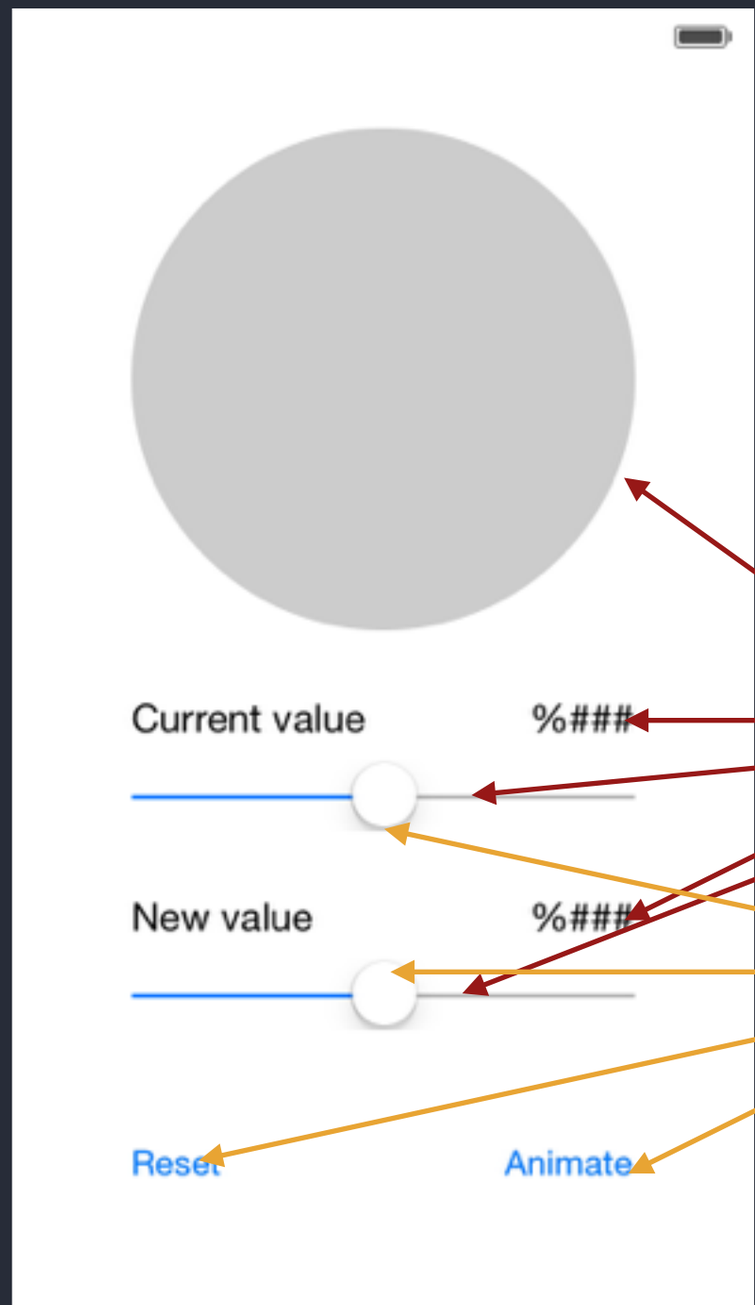








- Pronto, já podemos atrelar nossa interface gráfica com a ViewController.h




```
//
//  ViewController.h
//  Objective-C_IBDesignables&IBInspectables
//
//  Created by Kaê Angeli Coutinho on 2/26/15.
//  Copyright (c) 2015 Kaê Angeli Coutinho. All rights reserved.
//

#import <UIKit/UIKit.h>
#import "PieChartView.h"

@interface ViewController : UIViewController

// IBOutlets

@property (weak,nonatomic) IBOutlet PieChartView * pieChartView;
@property (weak,nonatomic) IBOutlet UILabel * pieChartCurrentValueLabel;
@property (weak,nonatomic) IBOutlet UISlider * pieChartCurrentValueSlider;
@property (weak,nonatomic) IBOutlet UILabel * pieChartNewValueLabel;
@property (weak,nonatomic) IBOutlet UISlider * pieChartNewValueSlider;

// IBActions

-(IBAction)changePieChartCurrentValue:(UISlider *)sender;
-(IBAction)changePieChartNewValue:(UISlider *)sender;
-(IBAction)resetPieChart:(UIButton *)sender;
-(IBAction)animatePieChart:(UIButton *)sender;

@end
```

IBOutlet
IBAction

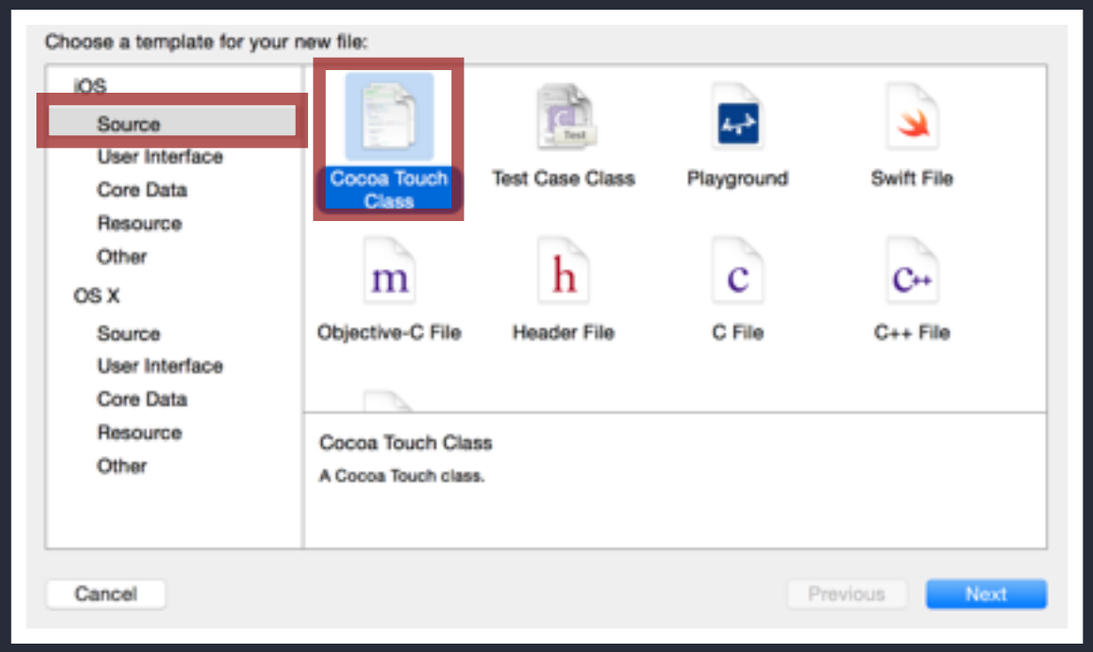
- Só o que nos resta é implementar a ViewController.m

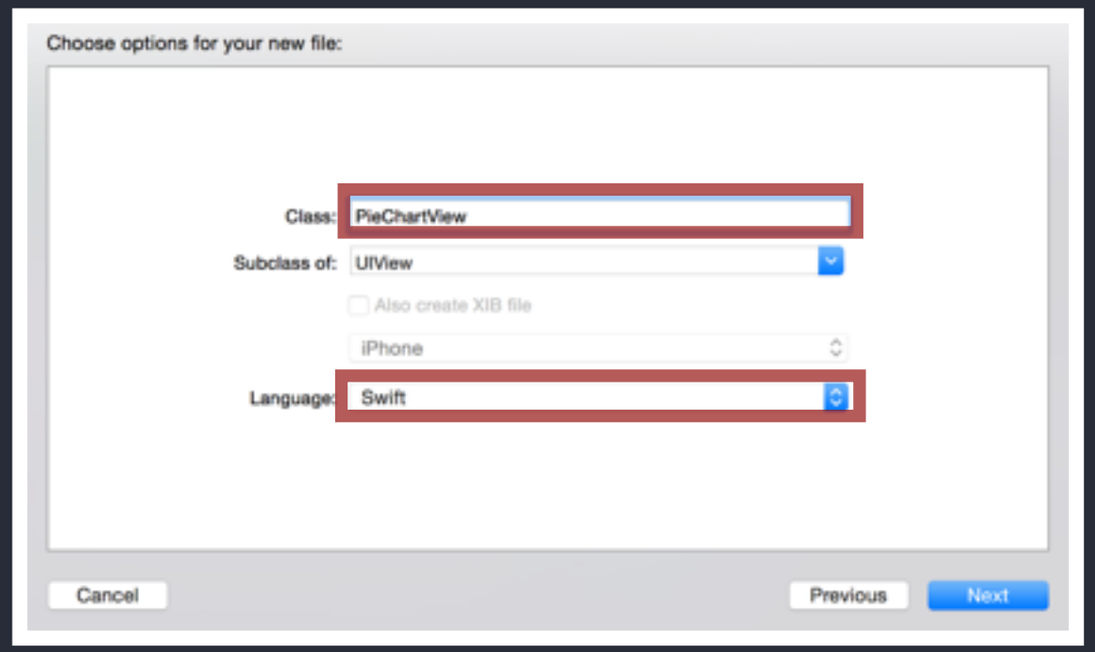
```
#pragma mark - UIView

-(void)viewDidLoad
{
    [super viewDidLoad];
    [self initialize];
}

-(void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
}
```

```
#pragma mark - Auxiliary Methods

-(void)initialize
{
    [[self pieChartView] setPiePercentage:0.0];
    [[self pieChartCurrentValueSlider] setValue:0.0];
    [[self pieChartNewValueSlider] setValue:0.0];
    [[self pieChartCurrentValueLabel] setText:[NSString stringWithFormat:@"%%%d",(int)([pieChartCurrentValueSlider value] * 100.0)]];
    [[self pieChartNewValueLabel] setText:[NSString stringWithFormat:@"%%%d",(int)([pieChartNewValueSlider value] * 100.0)]];
    [[self pieChartView] updateLayerProperties];
}
```

```
#pragma mark - IBActions

-(IBAction)changePieChartCurrentValue:(UISlider *)sender
{
    [[self pieChartCurrentValueLabel] setText:[NSString stringWithFormat:@"%%%d",(int)([sender value] * 100.0)]];
    [[self pieChartView] setPiePercentage:([sender value] * 100.0)];
    [[self pieChartView] updateLayerProperties];
}

-(IBAction)changePieChartNewValue:(UISlider *)sender
{
    [[self pieChartNewValueLabel] setText:[NSString stringWithFormat:@"%%%d",(int)([sender value] * 100.0)]];
}

-(IBAction)resetPieChart:(UIButton *)sender
{
    [self initialize];
}

-(IBAction)animatePieChart:(UIButton *)sender
{
    [[self pieChartView] updatePiePercentage:([[self pieChartNewValueSlider] value] * 100.0)];
    [[self pieChartCurrentValueLabel] setText:[NSString stringWithFormat:@"%%%d",(int)([[self pieChartNewValueSlider] value] * 100.0)]];
    [[self pieChartCurrentValueSlider] setValue:[[self pieChartNewValueSlider] value]];
    [[self pieChartView] setPiePercentage:(double)([[self pieChartNewValueSlider] value] * 100.0)];
}
```

- *Swift*
  - Começaremos com a criação da nova classe PieChartView





- Começaremos com a implementação da classe (PieChartView.swift)

```
@IBDesignable
class PieChartView: UIView
{
```

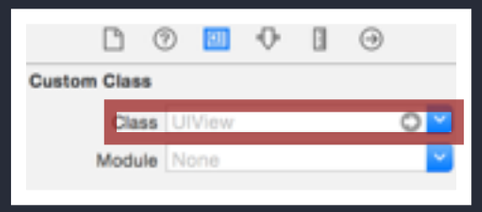
```
// MARK: Properties

var pieBackgroundLayer: CAShapeLayer!
var pieLayer: CAShapeLayer!
var pieStrokeWidth: Double = 0.0

@IBInspectable
var piePercentage: Double = 0.0

@IBInspectable
var pieBackgroundColor: UIColor = UIColor(white:0.8,alpha:1.0)

@IBInspectable
var pieColor: UIColor = UIColor.orangeColor()
```

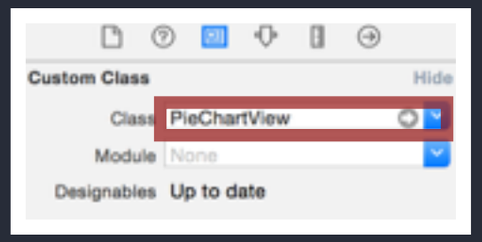
```
// MARK: Methods

override func layoutSubviews()
{
    super.layoutSubviews()
    self.pieStrokeWidth = Double(self.frame.size.width) / 2.0
    if !(self.pieBackgroundLayer != nil)
    {
        self.pieBackgroundLayer = CAShapeLayer()
        self.layer.addSublayer(self.pieBackgroundLayer)
        let circleRect = CGRectInset(self.bounds,
                                     CGFloat(self.pieStrokeWidth / 2.0),
                                     CGFloat(self.pieStrokeWidth / 2.0))
        let circlePath = UIBezierPath(ovalInRect:circleRect)
        self.pieBackgroundLayer.path = circlePath.CGPath
        self.pieBackgroundLayer.fillColor = nil
        self.pieBackgroundLayer.lineWidth = CGFloat(self.pieStrokeWidth)
        self.pieBackgroundLayer.strokeColor = self.pieBackgroundColor.CGColor
    }
    self.pieBackgroundLayer.frame = self.layer.bounds
    if !(self.pieLayer != nil)
    {
        self.pieLayer = CAShapeLayer()
        self.layer.addSublayer(self.pieLayer);
        let circleRect = CGRectInset(self.bounds,
                                     CGFloat(self.pieStrokeWidth / 2.0),
                                     CGFloat(self.pieStrokeWidth / 2.0))
        let circlePath = UIBezierPath(ovalInRect:circleRect)
        self.pieLayer.path = circlePath.CGPath
        self.pieLayer.fillColor = nil
        self.pieLayer.lineWidth = CGFloat(self.pieStrokeWidth)
        self.pieLayer.strokeColor = pieColor.CGColor
        self.pieLayer.anchorPoint = CGPointMake(0.5,0.5)
        self.pieLayer.transform = CATransform3DRotate(self.pieLayer.transform,CGFloat(-M_PI / 2.0),0.0,0.0,1.0)
    }
    self.pieLayer.frame = self.layer.bounds
    self.updateLayerProperties()
}
```

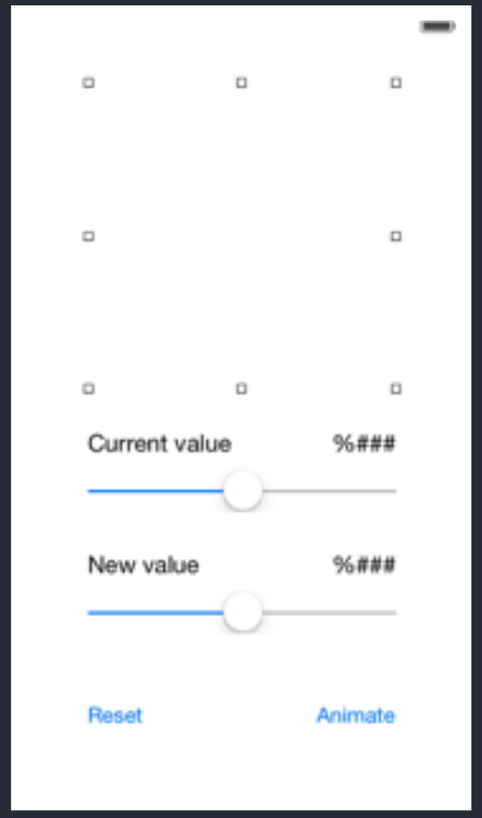
```
func updateLayerProperties()
{
    if (self.pieLayer != nil)
    {
        self.pieLayer.strokeEnd = CGFloat(self.piePercentage / 100.0)
    }
}
```

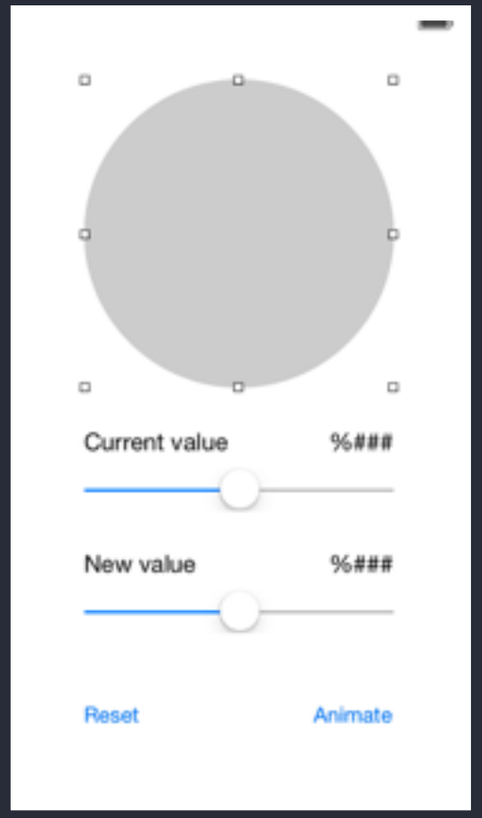
```
func updatePiePercentage(#newPiePercentage: Double)
{
    if (self.pieLayer != nil)
    {
        CATransaction.begin()
        var animation = CABasicAnimation(keyPath:"strokeEnd")
        animation.duration = ((newPiePercentage / 100.0) - (self.piePercentage / 100.0)) * 3.0
        animation.fromValue = self.piePercentage / 100.0
        animation.toValue = newPiePercentage / 100.0
        animation.timingFunction = CAMediaTimingFunction(name:kCAMediaTimingFunctionEaseInEaseOut)
        CATransaction.setCompletionBlock(
        { () -> Void in
            CATransaction.begin()
            CATransaction.setValue(kCFBooleanTrue,forKey:kCATransactionDisableActions)
            self.pieLayer.strokeEnd = CGFloat(newPiePercentage / 100.0)
            CATransaction.commit()
        })
        self.pieLayer.addAnimation(animation,forKey:"animateStrokeEnd")
        CATransaction.commit()
    }
}
```

- Iremos agora desenvolver nossa interface gráfica (storyboard)
- Reparem que nossa UIView não fora ainda renderizada pelo *Interface Builder*
- É necessário atrelar a classe do componente gráfico customizado recém-criado

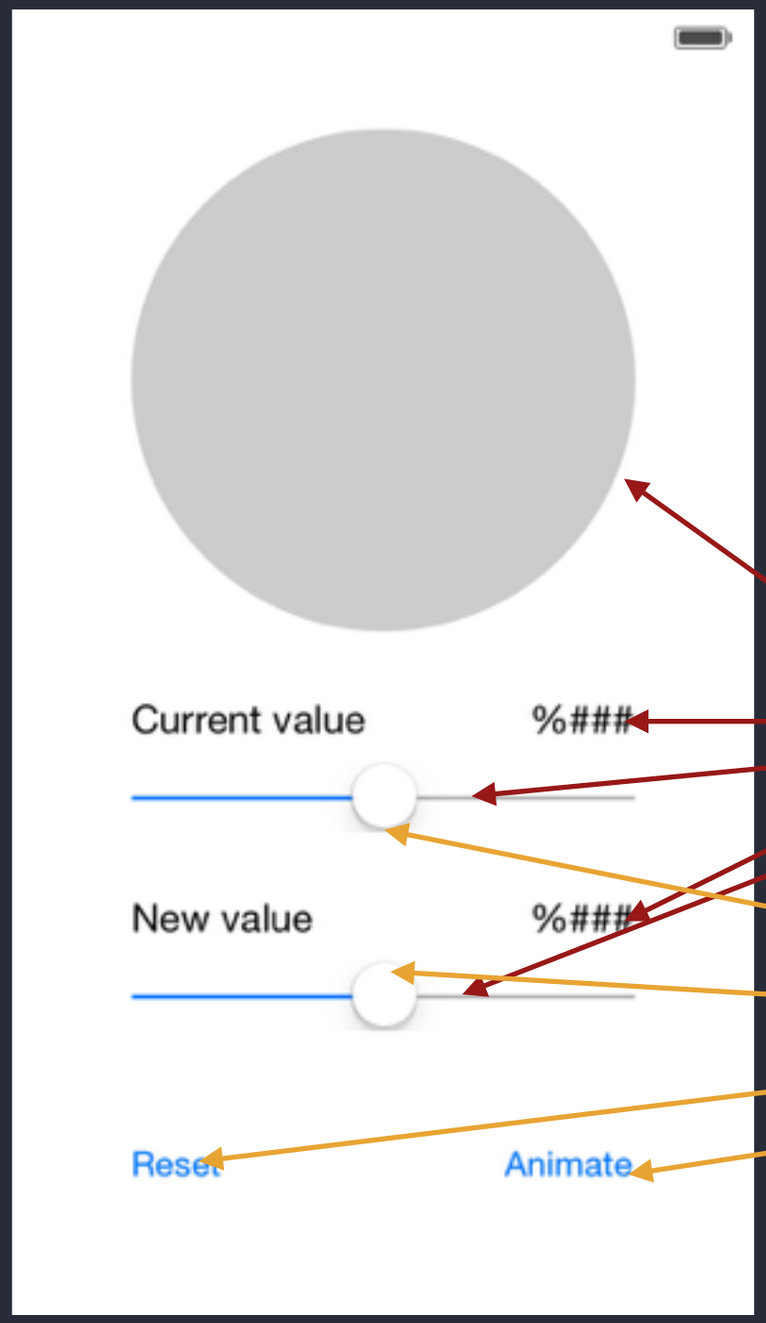








- Pronto, já podemos atrelar nossa interface gráfica com a ViewController.swift




```
1  //
2  //  ViewController.swift
3  //  Swift_IBDesignables&IBInspectables
4  //
5  //  Created by Kaê Angeli Coutinho on 2/26/15.
6  //  Copyright (c) 2014 Kaê Angeli Coutinho. All rights reserved.
7  //
8
9  import UIKit
10
11 class ViewController: UIViewController
12 {
13     // MARK: IBOutlets
14
15     @IBOutlet weak var pieChartView: PieChartView?
16     @IBOutlet weak var pieChartCurrentValueLabel: UILabel?
17     @IBOutlet weak var pieChartCurrentValueSlider: UISlider?
18     @IBOutlet weak var pieChartNewValueLabel: UILabel?
19     @IBOutlet weak var pieChartNewValueSlider: UISlider?
```

```
48  @IBAction func changePieChartCurrentValue(sender: UISlider)
55  @IBAction func changePieChartNewValue(sender: UISlider)
60  @IBAction func resetPieChart(sender: UIButton)
65  @IBAction func animatePieChart(sender: UIButton)
```

IBOutlet

IBAction

- Só o que nos resta é implementar a ViewController.swift

```
// MARK: UIView

override func viewDidLoad()
{
    super.viewDidLoad()
    self.initialize()
}

override func didReceiveMemoryWarning()
{
    super.didReceiveMemoryWarning()
}
```

```
// MARK: Auxiliary Methods

func initialize()
{
    self.pieChartView?.piePercentage = 0.0
    self.pieChartCurrentValueSlider?.value = 0.0
    self.pieChartNewValueSlider?.value = 0.0
    self.pieChartCurrentValueLabel?.text = NSString(format:"%%%d",Int(self.pieChartCurrentValueSlider!.value * 100.0))
    self.pieChartNewValueLabel?.text = NSString(format:"%%%d",Int(self.pieChartNewValueSlider!.value * 100.0))
    self.pieChartView?.updateLayerProperties()
}
```

```
// MARK: IBActions

@IBAction func changePieChartCurrentValue(sender: UISlider)
{
    self.pieChartCurrentValueLabel?.text = NSString(format:"%%%d",Int(sender.value * 100.0))
    self.pieChartView?.piePercentage = Double(sender.value * 100.0)
    self.pieChartView?.updateLayerProperties()
}

@IBAction func changePieChartNewValue(sender: UISlider)
{
    self.pieChartNewValueLabel?.text = NSString(format:"%%%d",Int(sender.value * 100.0))
}

@IBAction func resetPieChart(sender: UIButton)
{
    self.initialize()
}

@IBAction func animatePieChart(sender: UIButton)
{
    self.pieChartView?.updatePiePercentage(newPiePercentage:Double(self.pieChartNewValueSlider!.value * 100.0))
    self.pieChartCurrentValueLabel?.text = NSString(format:"%%%d",Int(self.pieChartNewValueSlider!.value * 100.0))
    self.pieChartCurrentValueSlider?.value = self.pieChartNewValueSlider!.value
    self.pieChartView?.piePercentage = Double(self.pieChartNewValueSlider!.value * 100.0)
}
```

Concluindo

- Neste tutorial, você aprendeu alguns conceitos importantes, como:
- Tópicos avançados do *Interface Builder*
- Nova sintaxe das linguagens *Objective-C* e *Swift*
- Conceitos básicos e avançados de *CoreGraphics* e *Quartz*