

# CRIANDO APPS PARA iOS

---



# BREVE HISTÓRICO

# PRIMÓRDIOS

Lançado como sistema operacional do 1º iPhone em 2007.

Originalmente chamado de iPhoneOS.

Era exclusivo para o dispositivo e totalmente fechado.

Continha apenas alguns Apps nativos e trazia uma experiência de uso diferenciada para a interação Multi-Touch capacitiva.

**Curiosidade:** ironicamente a Apple defendia nas primeiras versões do iOS que o futuro seriam os Web Apps, graças a experiência que eles conseguiram proporcionar com o Safari no primeiro iPhone.

# VERSÕES DE iOS

## iPhoneOS 1 (2007)

Lançamento do iPhone;  
Recursos básicos de Smartphone: ligação, SMS, Emails e Browser.

## iPhoneOS 2 (2008)

Lançamento da AppStore

## iOS 3 (2009)

Copy & Paste;  
Spotlight;  
In-App Purchases.

## iOS 4 (2010)

Multitasking;  
Suporte a Retina Display.

## iOS 8 (2014)

Extensões do App.

**Curiosidade:** o iOS 7 teve grande influência na tendência da indústria de software em adotar um design mais limpo e plano.

## iOS 9 (2015)

Multitarefa no i

Novas fontes do sistema;

Buscas nos Apps por Sp

Atualizações nos Ap  
nativos.

## iOS 7 (2013)

Novo design flat,  
abandonando o Esquemorismo;

Central de Controle;

Full Multitasking.

## iOS 5 (2011)

Central de Notificações;  
iCloud;  
Integração com redes sociais (Twitter).

## iOS 10 (2016)

**Curiosidade:** os releases do (Maps, Yo mudou a es

Remodelagem das Notificações, mais possibilidades de Interações

ativos do Google a ao Android, e partes do sistema.

## iOS 6 (2012)

Apple Maps;  
Diversas mudanças nos Apps nativos.

# FRAGMENTAÇÃO

A fragmentação de versões do iOS é bem menor do que no Android.

Graças ao controle total que a Apple tem sobre os dispositivos disponíveis para a plataforma no mercado.

# FRAGMENTAÇÃO

O ciclo de releases, desde a versão 4.0, tornou-se previsível

- 1 Major release por ano

- A nova versão é apresentada no WWDC (conferência anual da Apple para desenvolvedores)

- Nele a Apple apresenta todas as novidades da nova versão para os desenvolvedores

- Um período de Beta testes acontece durante 3 a 4 meses (geralmente entre junho e setembro)

- Até 4 minor releases até fechar o ciclo da versão

# FRAGMENTAÇÃO

Cada nova versão deixa de suportar dispositivos mais antigos, sendo que alguns dispositivos tiveram ciclos de atualização maiores.

# DISPOSITIVOS



## iPhone

Principal dispositivo suportado pelo iOS

Lança as novas tecnologias que serão depois adaptadas a outros devices...

Multi-Touch Capacitivo

Telas Retina

Touch ID

3D Touch

Câmera Dupla com Zoom Óptico

# DISPOSITIVOS

## iPad

Tablets da Apple

Atualmente em 3 tamanhos:

Mini (7,9')

Air (9,7')

Pro (12,9')

Processadores mais poderosos e  
mais memória que os iPhones.

Suportam algumas funções  
específicas como Split-View e PIP

Acessórios específicos como Apple  
Pen



# DISPOSITIVOS



## iPod Touch

Um iPhone sem a função de telefone.

Ciclo de releases mais lento.

Tende a ser descontinuado pela Apple.

Comumente usado como dispositivo de testes por ser mais 'barato'.

# DISPOSITIVOS

## Apple TV

Dispositivo para levar funções interativas e de mídia as TV's.

Até a 3<sup>a</sup> geração usava uma versão modificada do iOS.

Interface otimizada para TV

Não tinha loja de Aplicativos

A partir de 4<sup>a</sup> geração foi redesenhada e introduziu o TvOS.

Pode projetar a tela de um iPhone ou iPad através do AirPlay.



# DISPOSITIVOS



## Apple Watch

Smartwatch da Apple

Utiliza o WatchOS que é derivado  
do iOS

Depende de um iPhone para  
funcionar

Seus Apps interagem diretamente  
com o iPhone em que esta  
sincronizado



# ARQUITETURA E RECURSOS

# ARQUITETURA

## Derivado do OSX

O iOS foi adaptado a partir do código do OSX

Portanto também da família UNIX

Utiliza o núcleo Darwin/BSD

O iOS é otimizado para a arquitetura de processadores ARM

# ARQUITETURA

## Apps executam no modelo 'Sandbox'

O App executa num ambiente controlado, uma 'caixa de areia'

Os Apps não tem conhecimento e não podem acessar os processos de outros Apps ou se comunicar diretamente com eles

Inibe a ação de Apps nocivos já que eles não podem prejudicar outras partes do sistema

# ARQUITETURA

Proporciona serviços para acessar recursos e sensores presentes nos dispositivos

Serviço de Localização (GPS)

Acelerometro/Giroscópio

Camera

Microfone

Implementa um modelo de segurança e privacidade baseado em permissões

# ARQUITETURA

Implementa um modelo de segurança e privacidade baseado em permissões

Modelo rígido de distribuição de Apps

Somente Apps assinados com certificados digitais válidos podem ser instalados

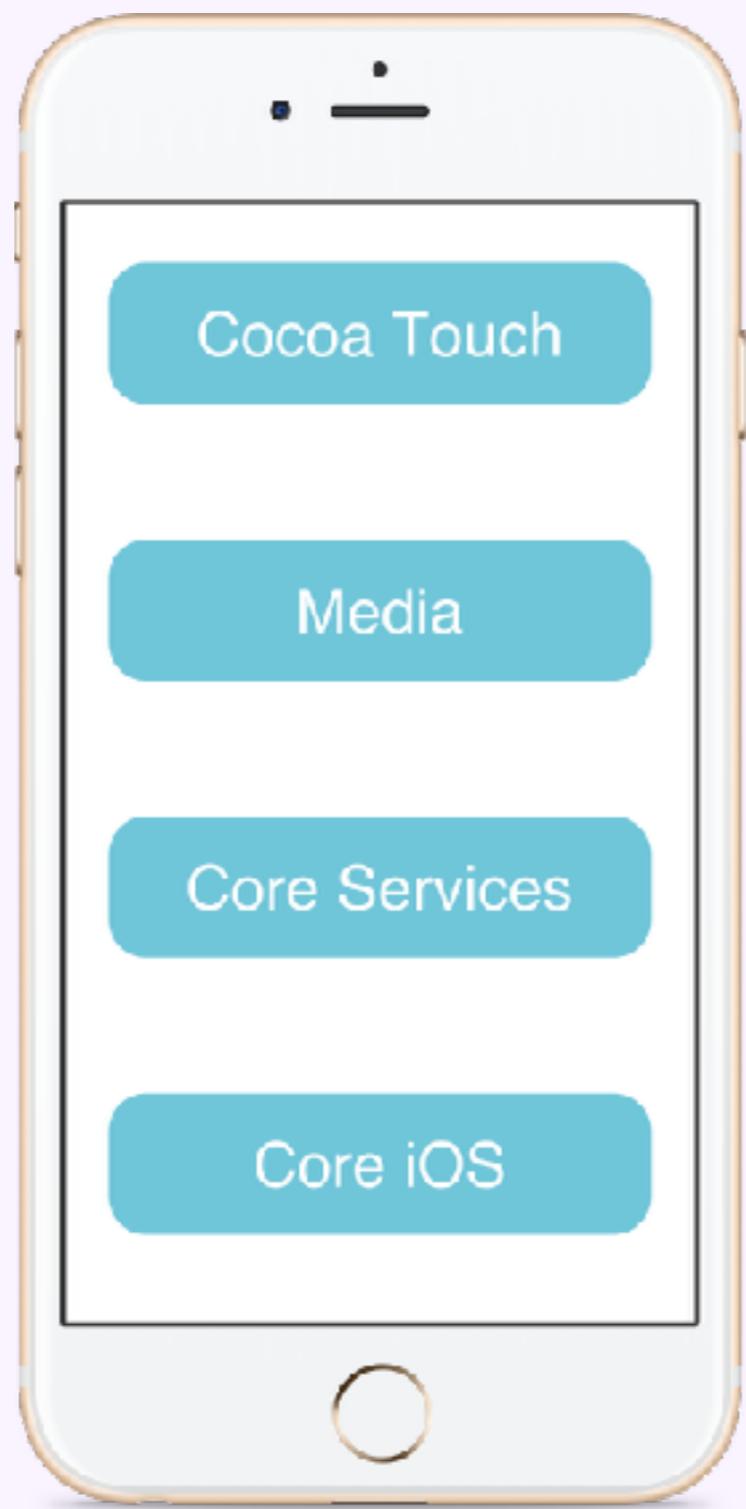
Usuários comuns somente podem instalar seus Apps a partir da App Store

# RECURSOS DA PLATAFORMA

A plataforma iOS é dividida em diversas camadas oferecendo os recursos necessários para desenvolvimento de Apps.

As camadas mais baixas estão mais próximas do hardware enquanto as superiores implementam níveis de abstração que facilitam a implementação de funcionalidades comuns.

# PRINCIPAIS CAMADAS



# CORE iOS

Acessam os componentes e serviços fundamentais do Sistema Operacional.

## Componentes

Kernel do OSX

Gerenciamento de Energia (Power Management)

Mach 3.0

Keychain Access

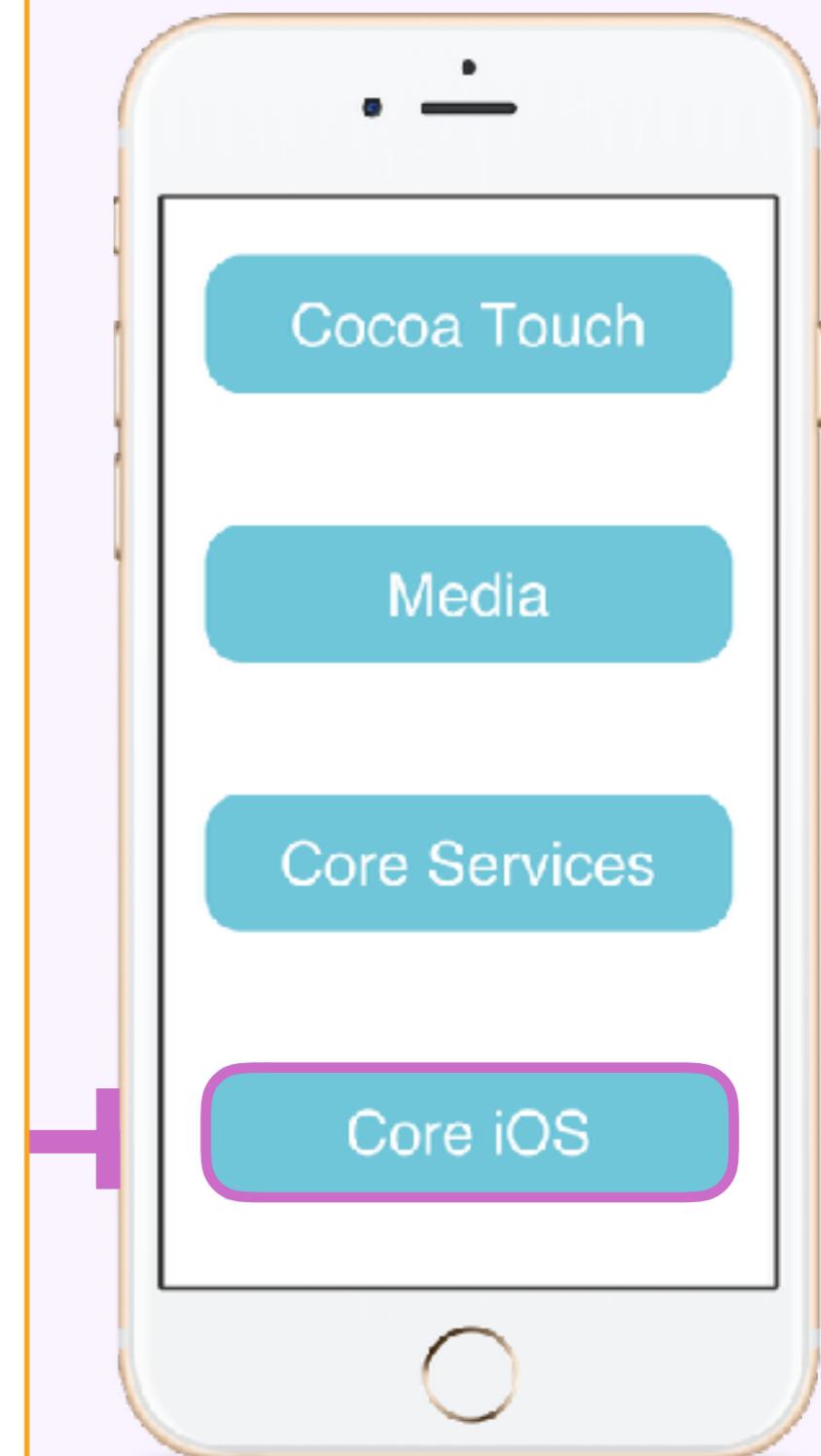
Certificados

Sockets

Sistema de arquivos

Segurança

Bonjour



# CORE SERVICES

Bibliotecas e serviços

Componentes

Coleções (Collection)

Localização Geográfica (Core Location)

Contatos (Address Book)

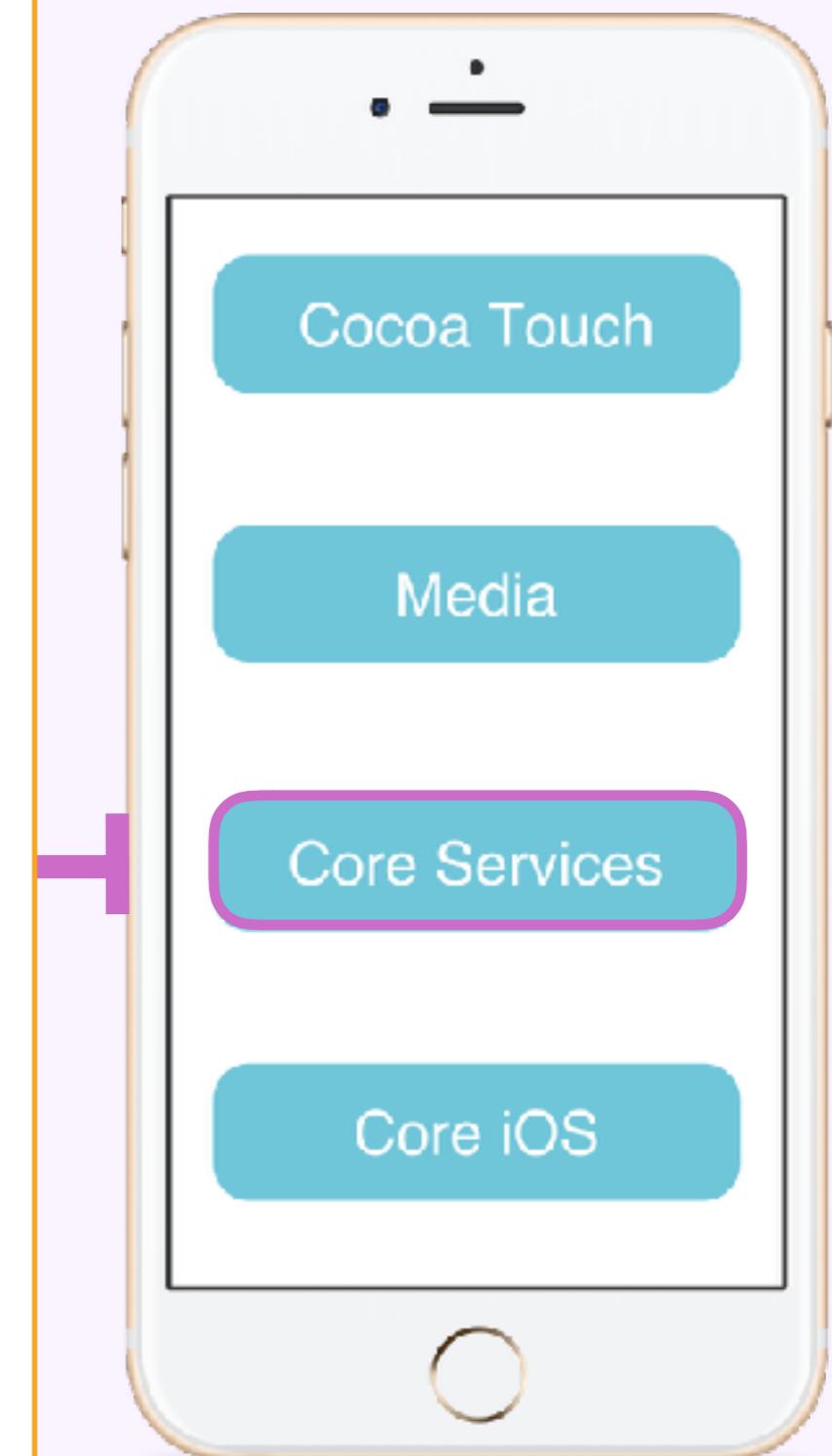
Serviços de rede

Threading

Acesso a arquivos

Preferências

SQLite



# MEDIA

Bibliotecas e serviços para reprodução e manipulação de mídia

## Componentes

Reprodução de Áudio (Core Audio)

OpenAL

JPEG, PNG, TIFF

PDF

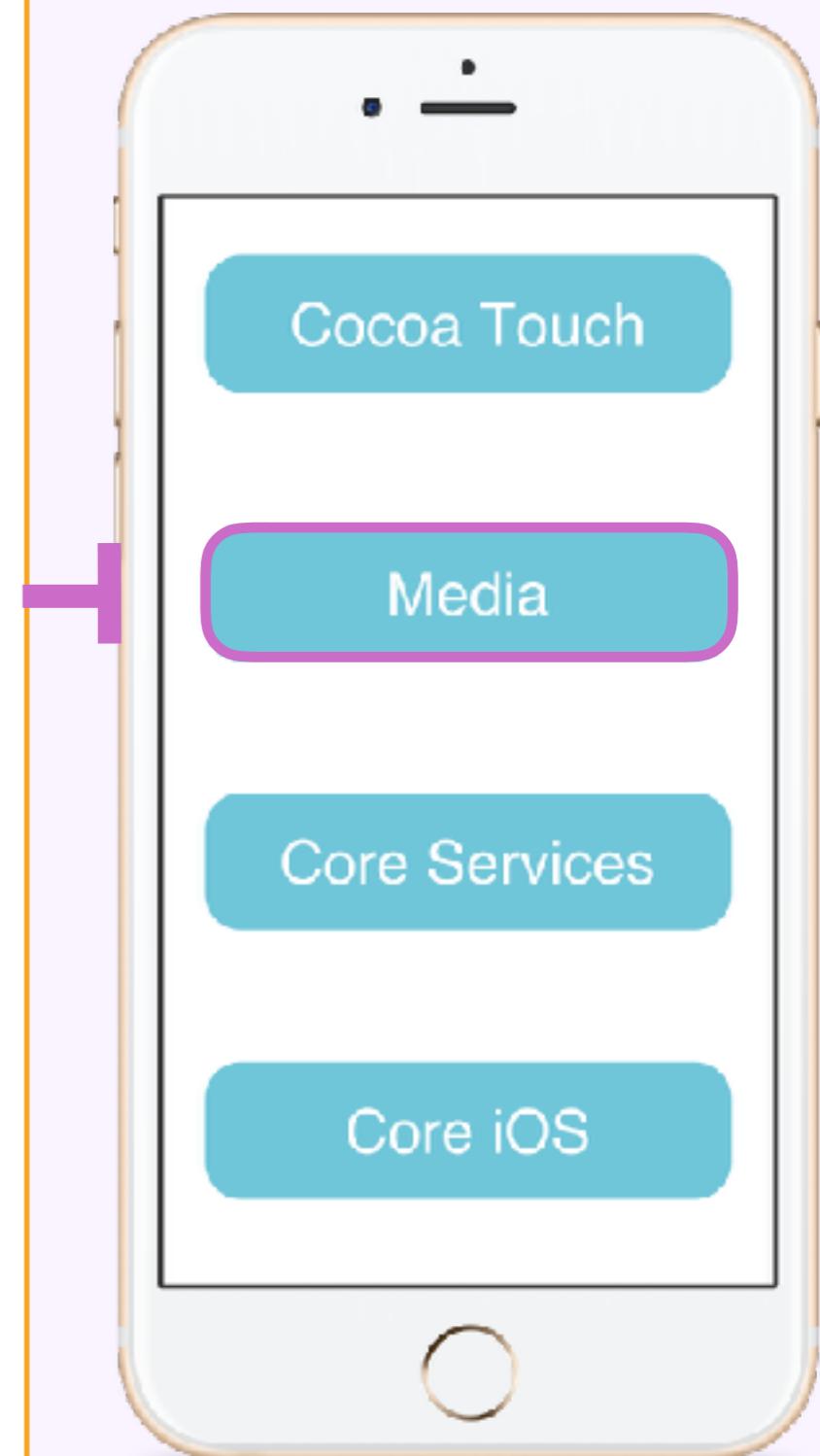
Gravação/Mixagem de Áudio

Desenhos e animações 2D (Quartz)

Animação (Core Animation)

Reprodução de Vídeo

OpenGL ES

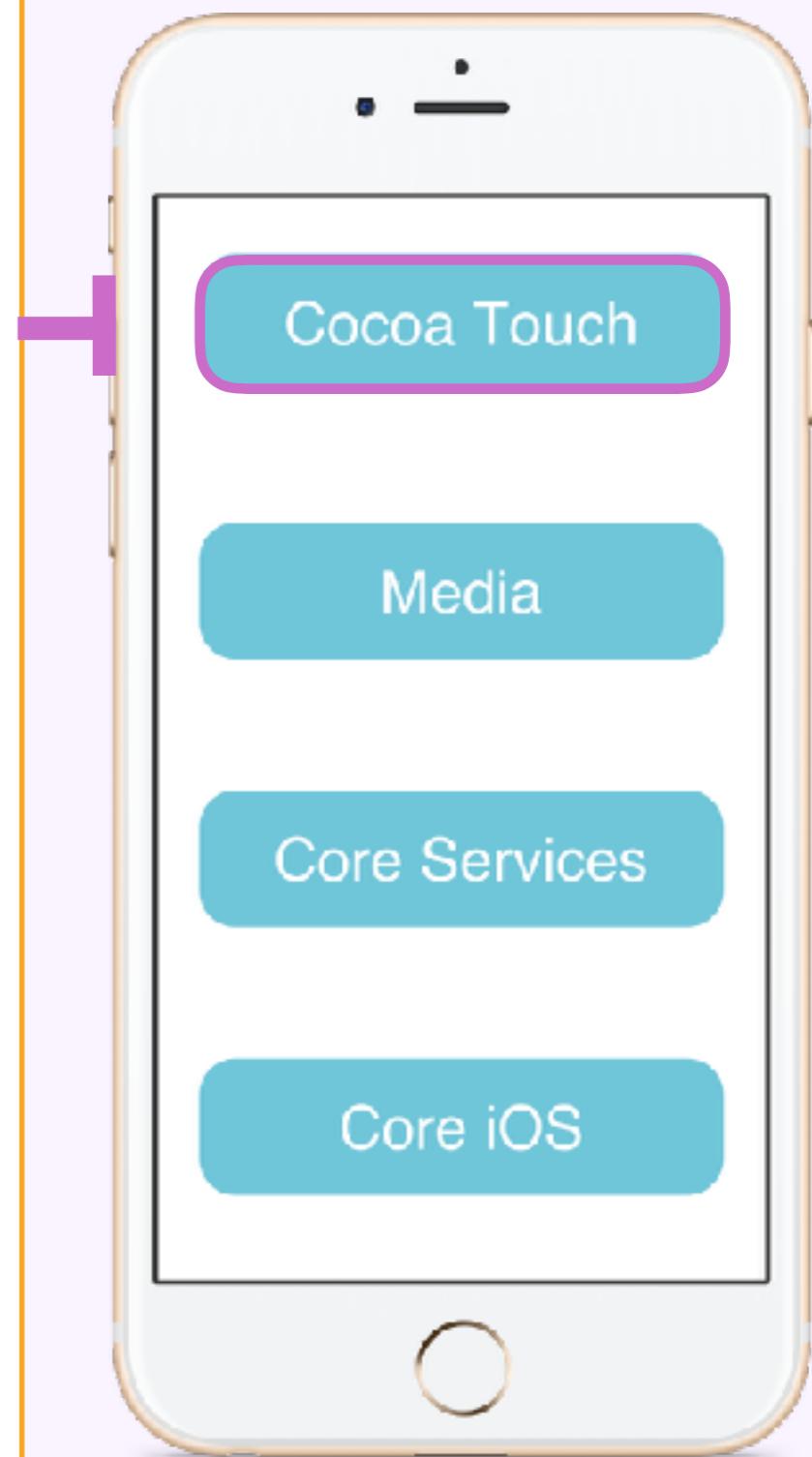


# COCOA TOUCH

Bibliotecas de mais alto nível na plataforma do iOS.

Possibilita a criação de interfaces, processamento de entrada e acesso alto nível aos sensores e serviços do sistema.

**Observação:** nosso treinamento é focado nessa camada



# COCOA TOUCH

## Componentes

Views e View Controllers

Controles

Reconhecedores de Gestos (Gesture Recognizers)

Alertas

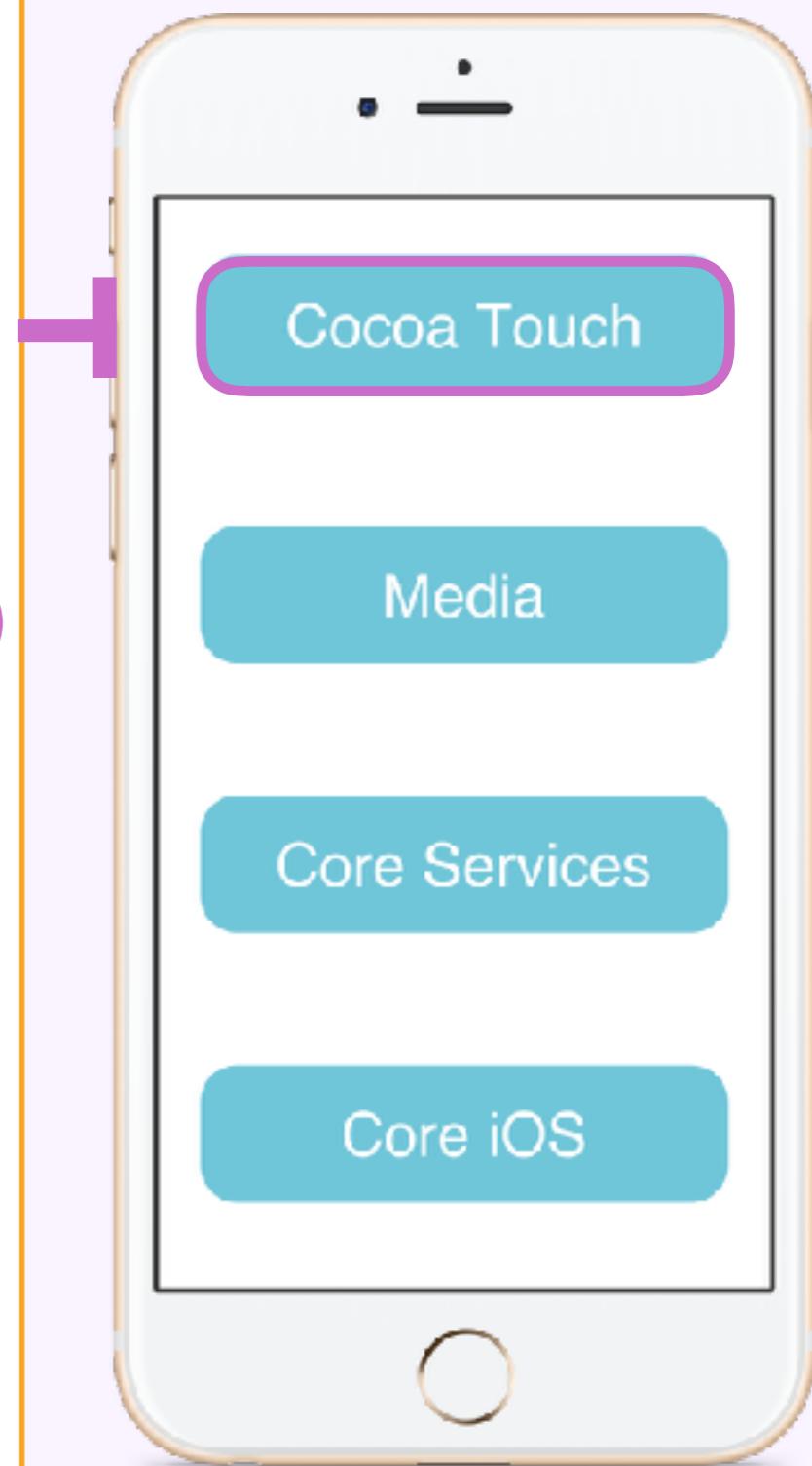
Localização e Internacionalização

MapKit

WebView

Image Picker

**Observação:** nosso treinamento  
é focado nessa camada





# COMPONENTES DA PLATAFORMA

# COMPONENTES DA PLATAFORMA

## Ferramentas



Xcode

Instruments

Oferecidas gratuitamente pela Apple, porém disponíveis apenas para Mac

## Linguagens

### Objective-C



Swift

## Frameworks (Bibliotecas)

### Core Data

### MapKit

### UIKit

### Foundation

...

## Estratégias de Design

### MVC



# O PATTERN MVC

# O PATTERN MVC

Pattern fundamental na modelagem de aplicações  
Cocoa e Cocoa Touch

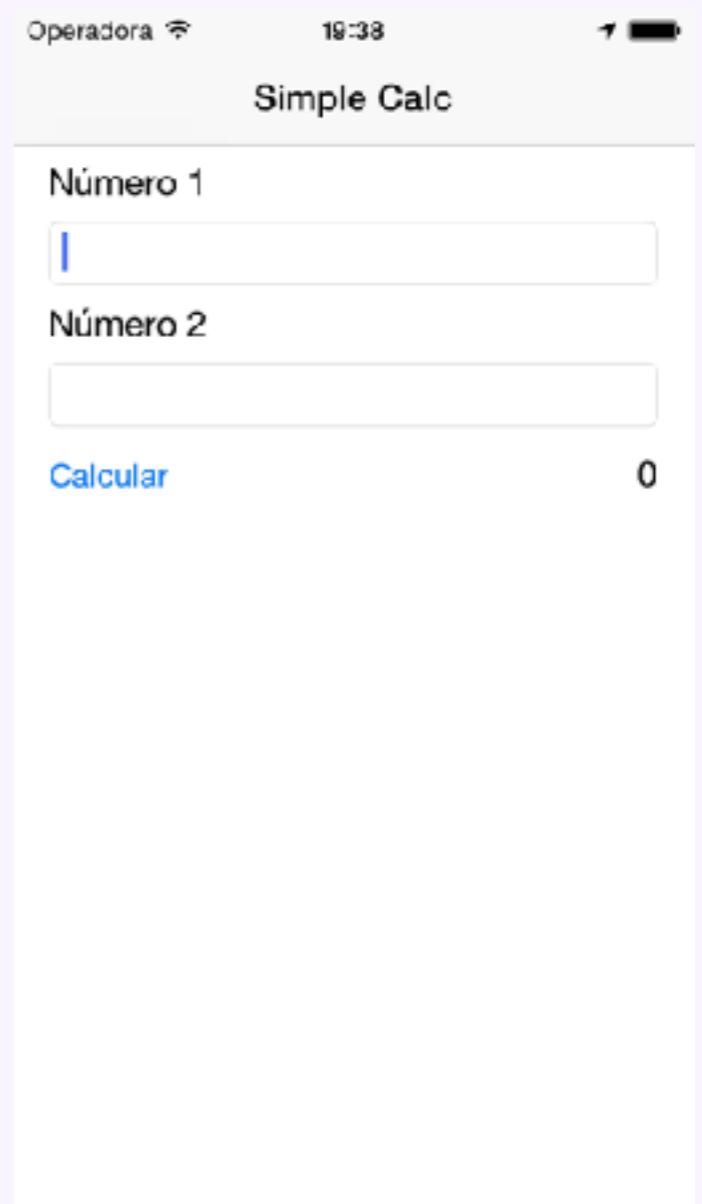
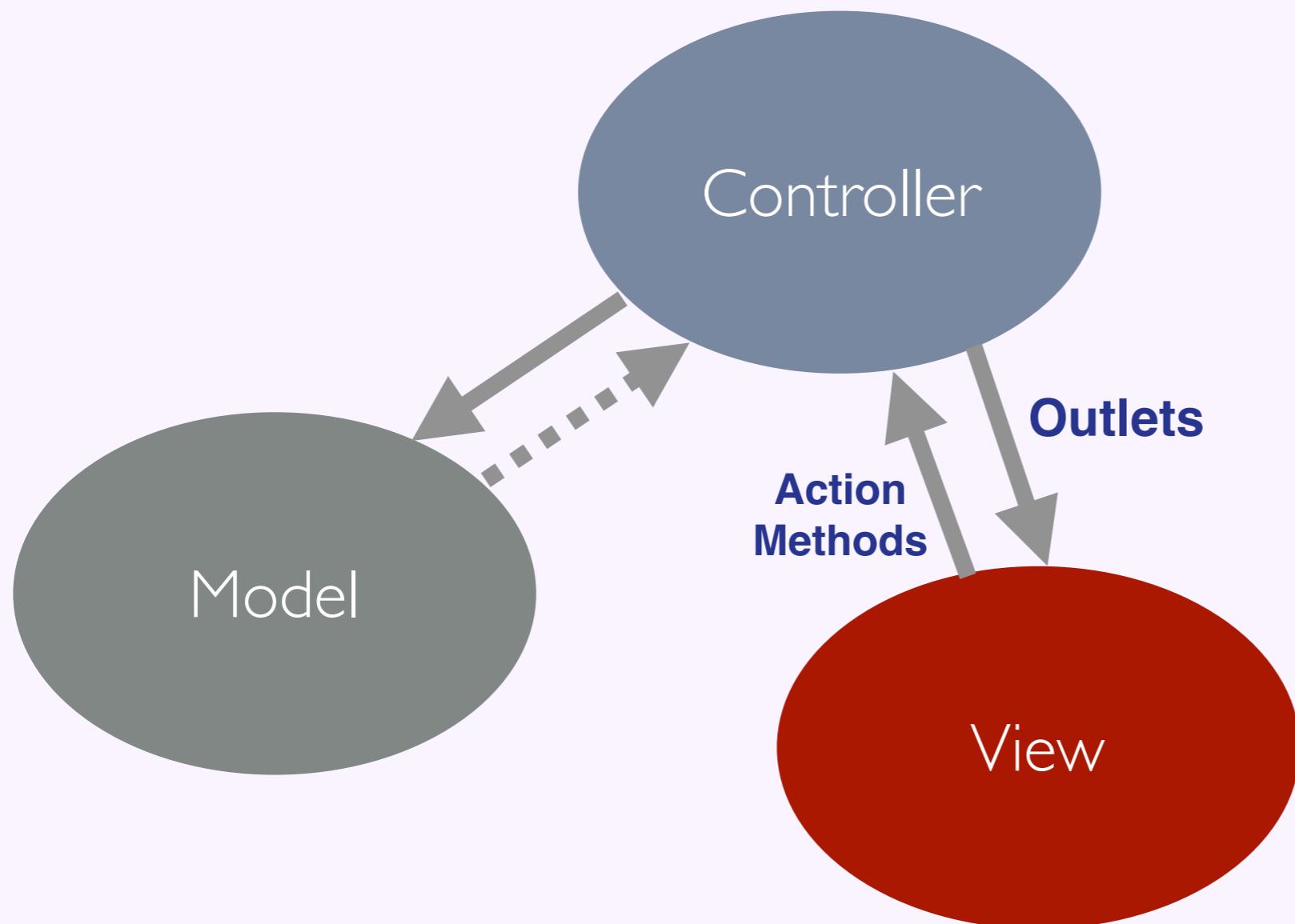
Divide a construção de uma aplicação em 3 partes fundamentais:

**Model:** responsável por toda a parte lógica e estrutural da aplicação, o seu “cérebro”.

**View:** responsável por apresentar a interface da aplicação, da maneira mais adequada a sua proposta.

**Controller:** responsável por gerenciar a apresentação da interface da aplicação e sua interação com o modelo.

# O PATTERN MVC



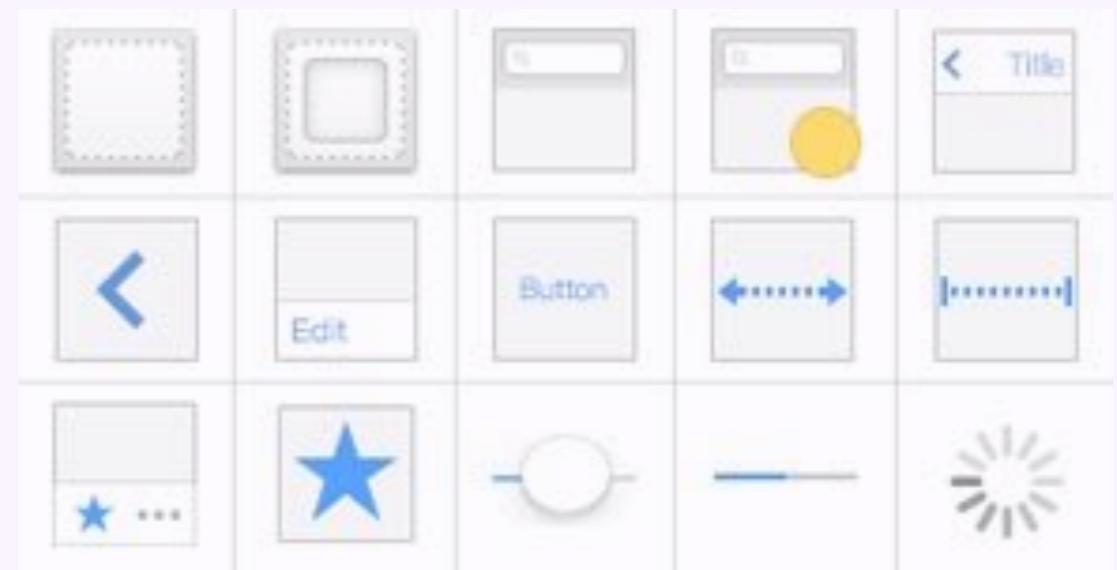


# APRESENTANDO O UIKIT

# APRESENTANDO O UIKIT

Framework fundamental  
para a construção e  
gerenciamento de Apps de  
iOS.

Proporciona a arquitetura  
para construção da  
interface de usuário do  
App, bem como suas  
interações.

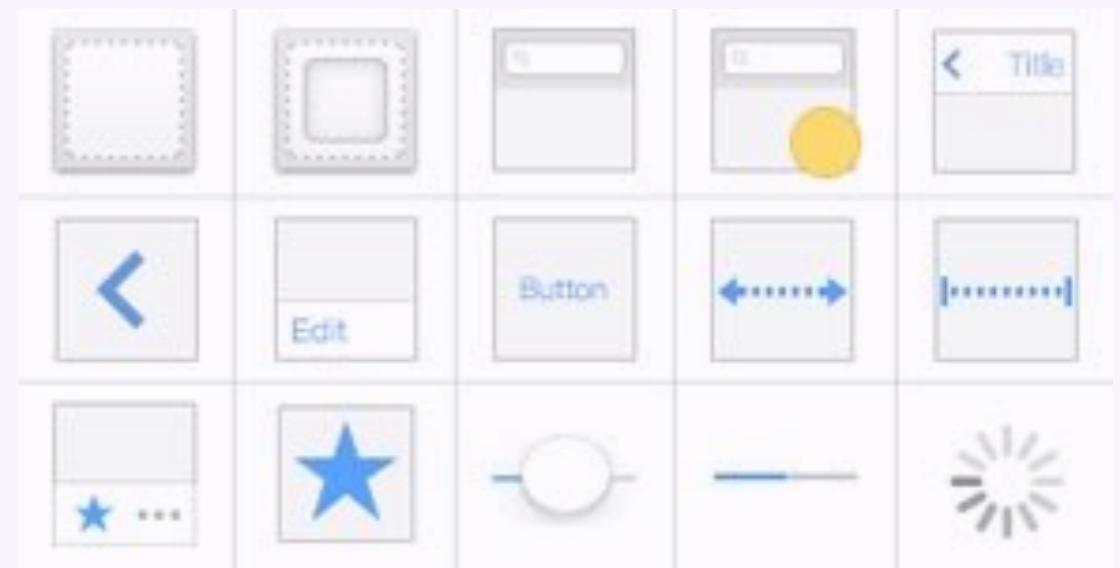


# PRINCIPAIS FUNCIONALIDADES

Gerenciamento da execução  
do App no plano principal ou  
no plano de fundo.

Modelo de *View Controllers*  
para gerenciamento de  
interface.

Conjunto rico de  
componentes visuais  
personalizáveis.

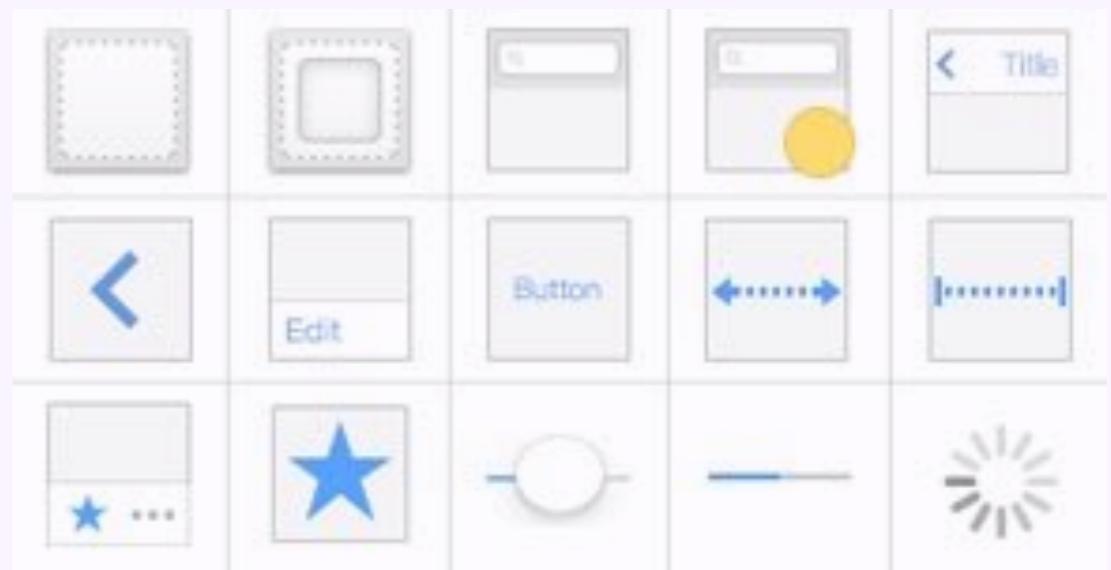


# PRINCIPAIS FUNCIONALIDADES

Animações dos elementos visuais. Manipulação e tratamento de toques e gestos.

*Pasteboard* - operações de copiar, cortar e colar.

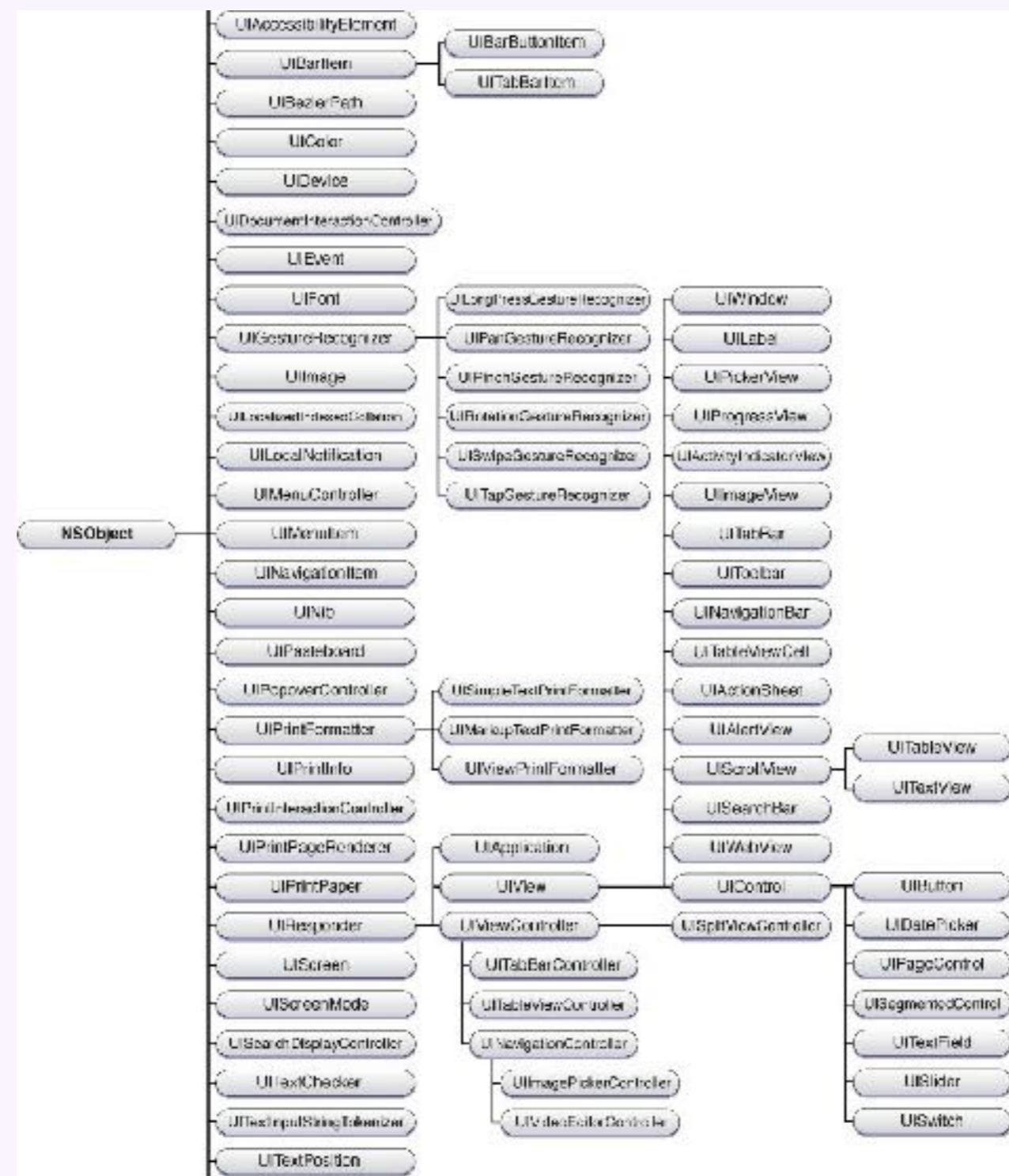
Acesso às câmeras e à biblioteca de imagens do usuário.

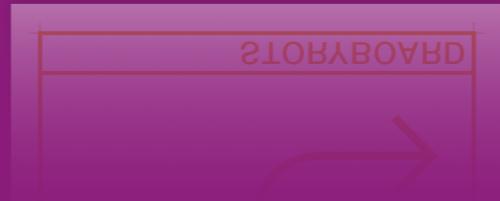
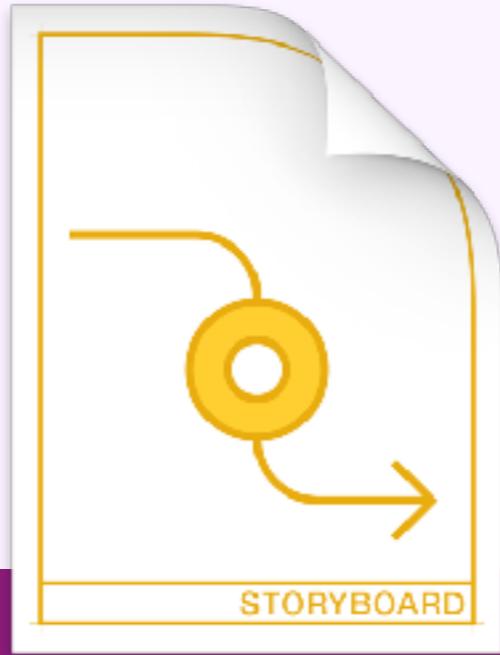




CLASSES DO UIKIT

# CLASSES DO UIKIT

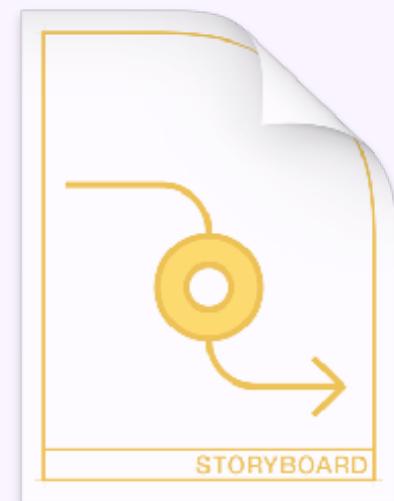




# OUTLETS E ACTION METHODS

# OUTLETS E ACTION METHODS

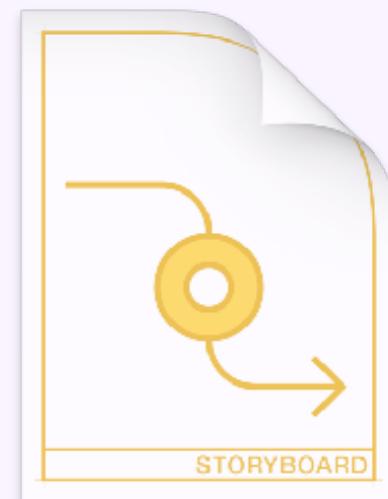
**Outlets** são conexões entre um componente desenhado no **View Controller** do Storyboard e uma propriedade no código fonte do **Controller**, permitindo que o **View Controller** interaja com os componentes da tela.

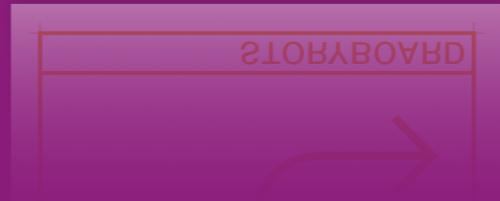
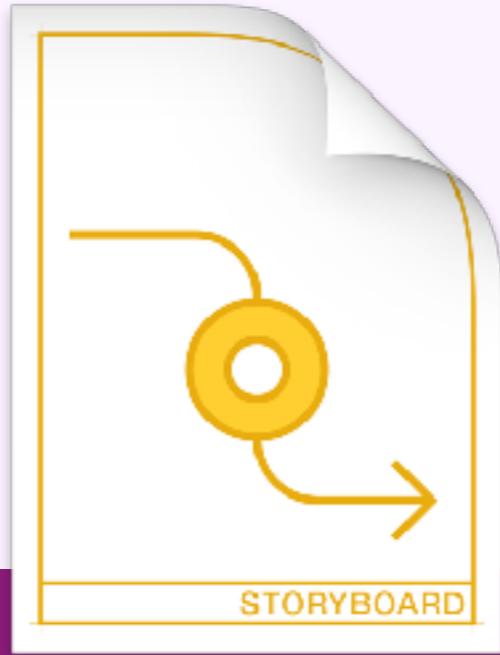


# OUTLETS E ACTION METHODS

*Action Methods* são métodos disparados em resposta aos eventos gerados pelos componentes da tela, como o toque em um botão ou a seleção de um novo valor em um *Slider*.

Ambos são criados graficamente usando o *Interface Builder* e o *Assistant Editor*.



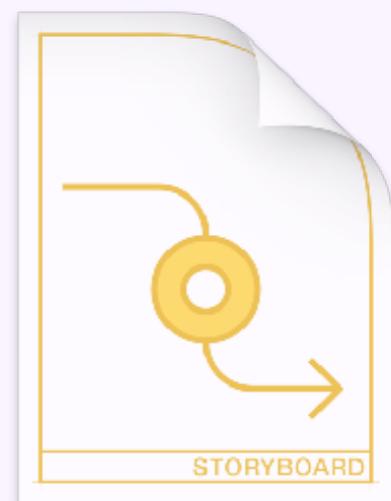


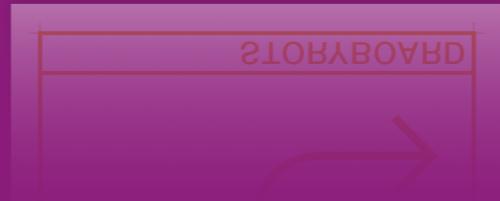
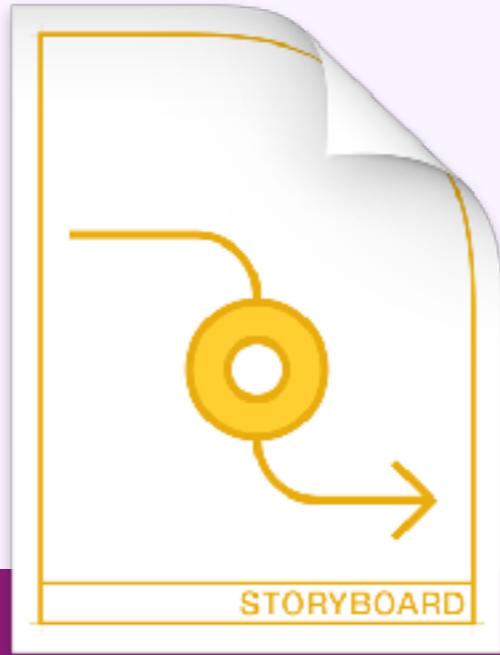
# AUTO-LAYOUT

# AUTO LAYOUT

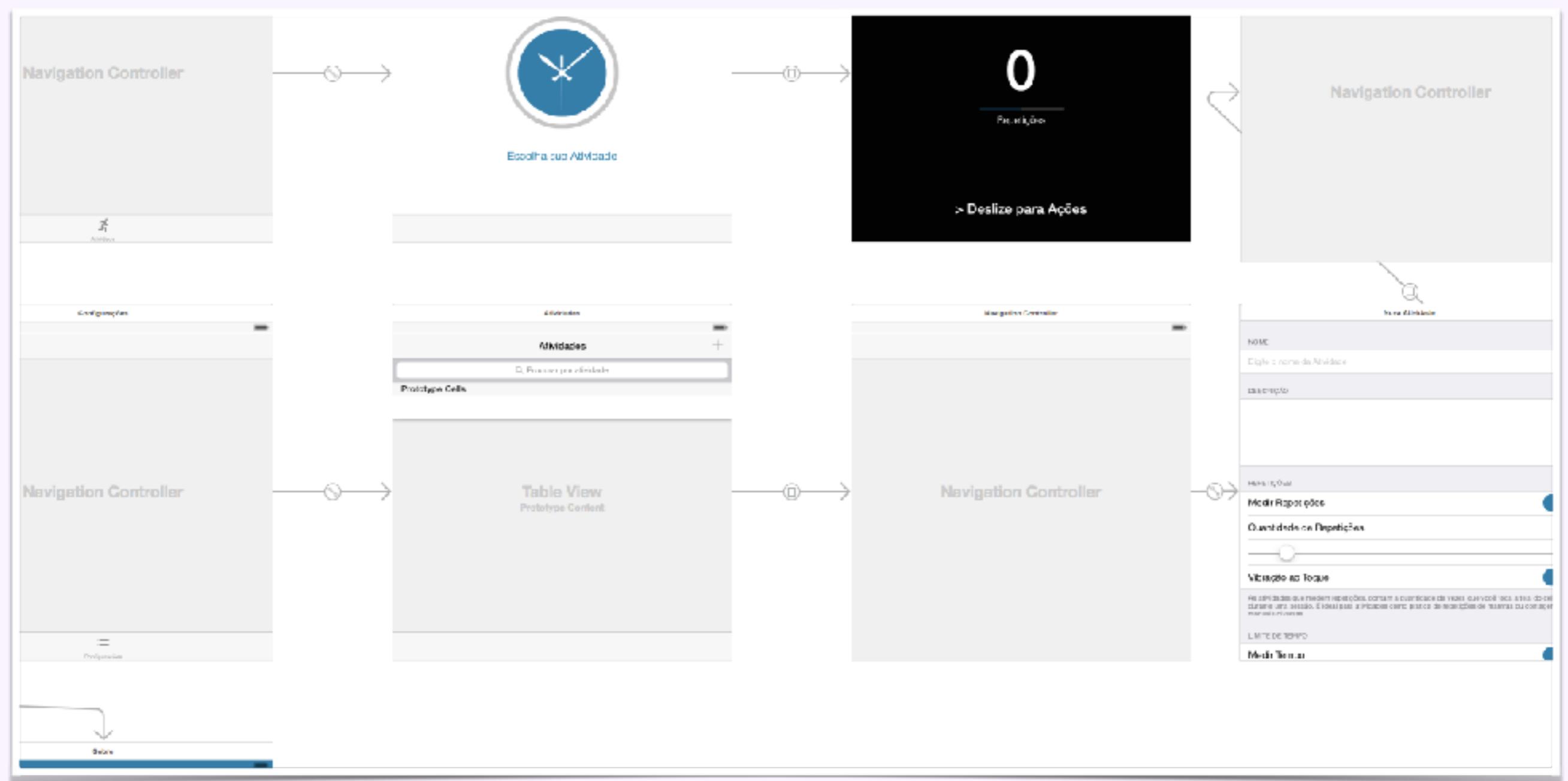
Um Framework poderoso para descrever o posicionamento e as dimensões dos elementos da tela.

Permite criar facilmente interfaces adaptáveis aos diversos tamanhos e resoluções de tela dos dispositivos.





# APRESENTANDO O STORYBOARD

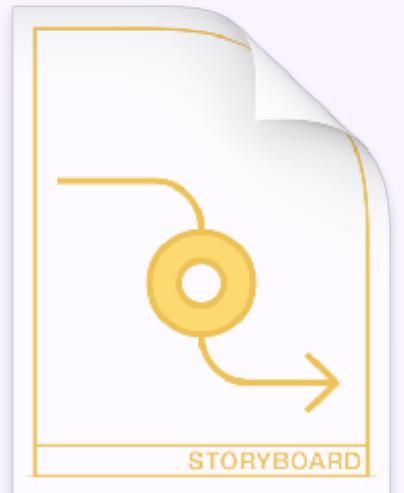


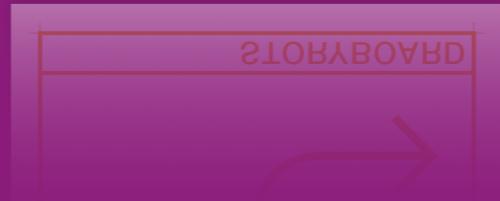
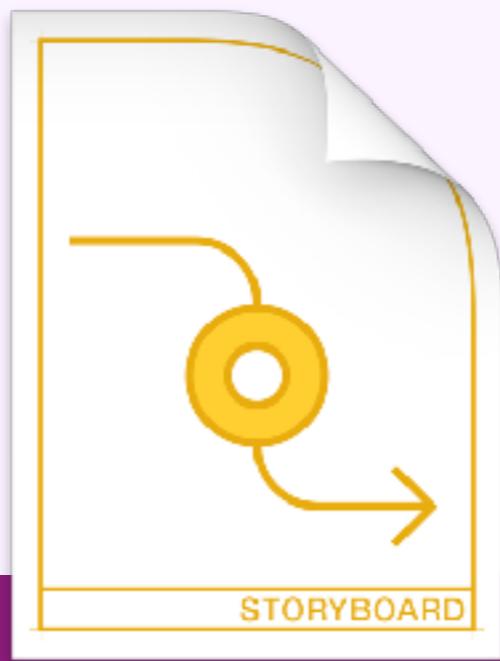
# APRESENTANDO O STORYBOARD

Agrupa os *Views Controllers* do App em um único documento.

Descreve os relacionamentos e navegação entre os *View Controllers* e outros componentes.

Interface única para construção das telas do App, usando um editor visual rico e uma extensa biblioteca de componentes.





# INTERFACE BUILDER

# INTERFACE BUILDER

Ferramenta gráfico para edição do Storyboard.

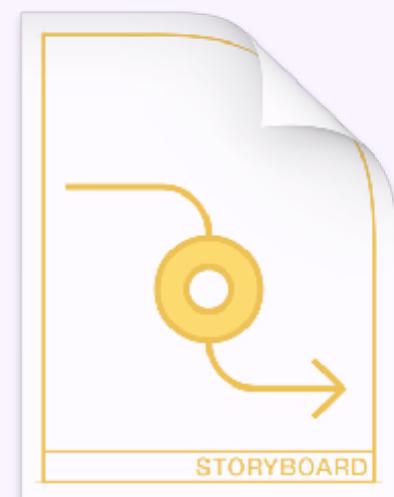
Funcionalidades:

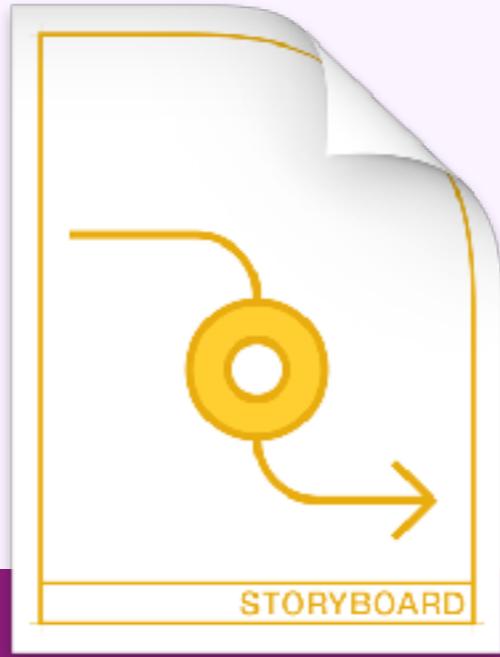
Visualizar e editar um Storyboard

Desenhar a Interface de uma tela

Criar conexões entre os elementos de interface

Criar conexões com o código fonte para criar Outlets e Action Methods, possibilitando a interação da interface com o código e vice-versa





# VIEW CONTROLLERS

# VIEW CONTROLLERS

Representa uma tela ou a porção de uma tela que é gerenciada por uma classe que herda *UIViewController*.

É a unidade básica de edição do Storyboard.

Onde os componentes são desenhados para compor a interface com o usuário.



# CICLO DE VIDA DO VIEW CONTROLLER

Sequência de eventos que acontecem no ciclo de vida de um *View Controller*:

`awakeFromNib` - chamado quando o View Controller é criado, adequado para lógica de inicialização da classe.

`viewDidLoad` - chamado quando o View Controller é carregado a partir do Storyboard. Adequado para configurar o View Controller e seus componentes.



# CICLO DE VIDA DO VIEW CONTROLLER

Sequência de eventos que acontecem no ciclo de vida de um *View Controller*:

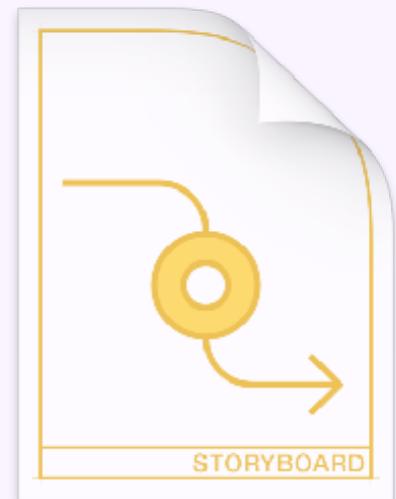
`viewWillAppear/viewDidAppear` - é chamado logo antes da View ser exibida na tela e imediatamente depois. Pode ser usado para sincronizar os componentes da tela com o modelo.



# CICLO DE VIDA DO VIEW CONTROLLER

Sequência de eventos que acontecem no ciclo de vida de um *View Controller*:

`viewWillLayoutSubviews/`  
`viewDidLayoutSubviews` - é chamado sempre que o tamanho da View está para mudar ou acabou de mudar, permitindo ao View Controller adequar a disposição dos componentes na View.

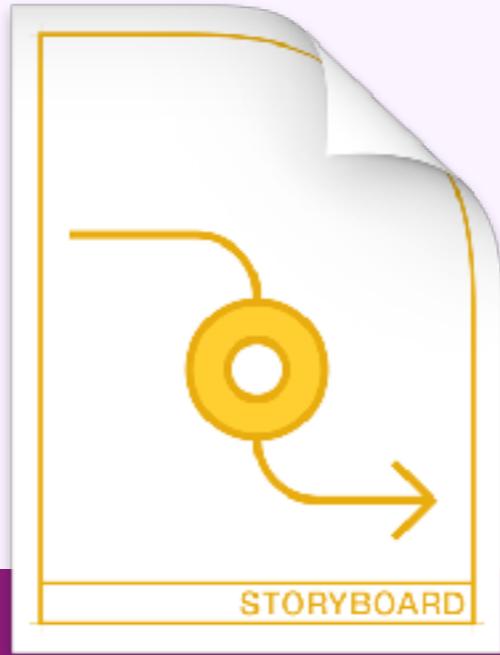


# CICLO DE VIDA DO VIEW CONTROLLER

Sequência de eventos que acontecem no ciclo de vida de um *View Controller*:

`viewWillDisappear/viewDidDisappear` - é chamado logo antes da View ser removida tela e imediatamente depois. Pode ser usada para limpar recursos que ocupem muita memória ou processamento e que possam ser recriados quando a View for re-exibida.





# SEGUES E MODOS DE APRESENTAÇÃO

# SEGUES E MODOS DE APRESENTAÇÃO

Segues conectam dois *View Controllers* dentro do Interface.

Configuram um modo de apresentação que descreve uma transição ou um composição de *View Controllers*.

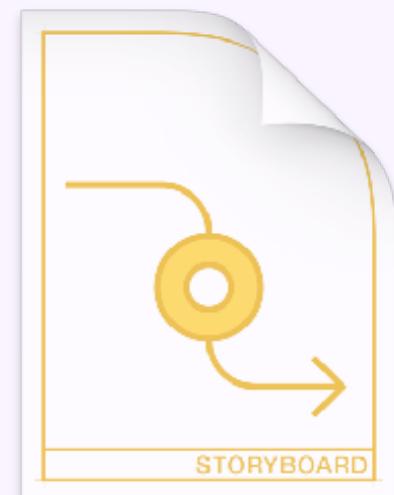
Principais modos de apresentação:

Show

Show Detail

Present Modally

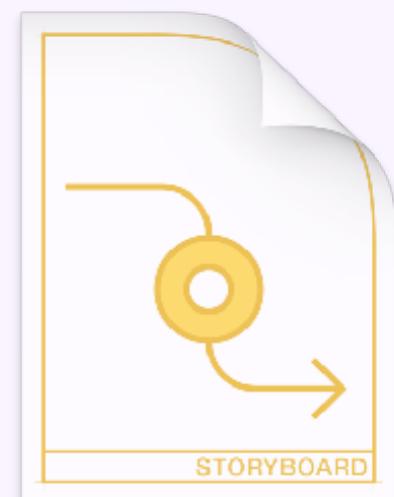
Present as Popover



# SEGUES E MODOS DE APRESENTAÇÃO

Atributo **Identifier** permite configurar um nome pelo qual podemos identificar no View Controller.

Comunicação entre View Controllers conectados é feita através do método ***prepareForSegue:identifier:***.

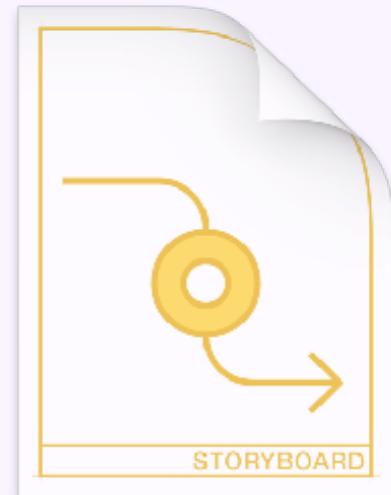


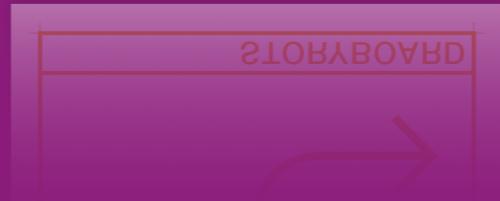
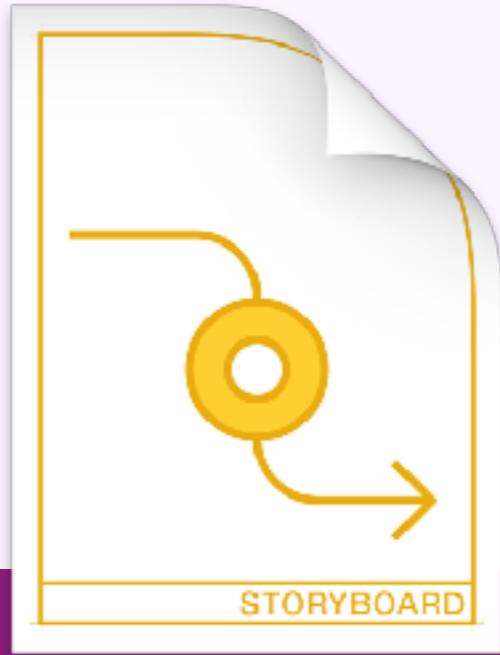
# SEGUES E MODOS DE APRESENTAÇÃO

Segues de transição podem usar animações padrões ou personalizadas pelo desenvolvedor.

Segues de composição permitem criar telas compostas por 2 ou mais View Controllers.

*Ex: Tab Bar Controller.*





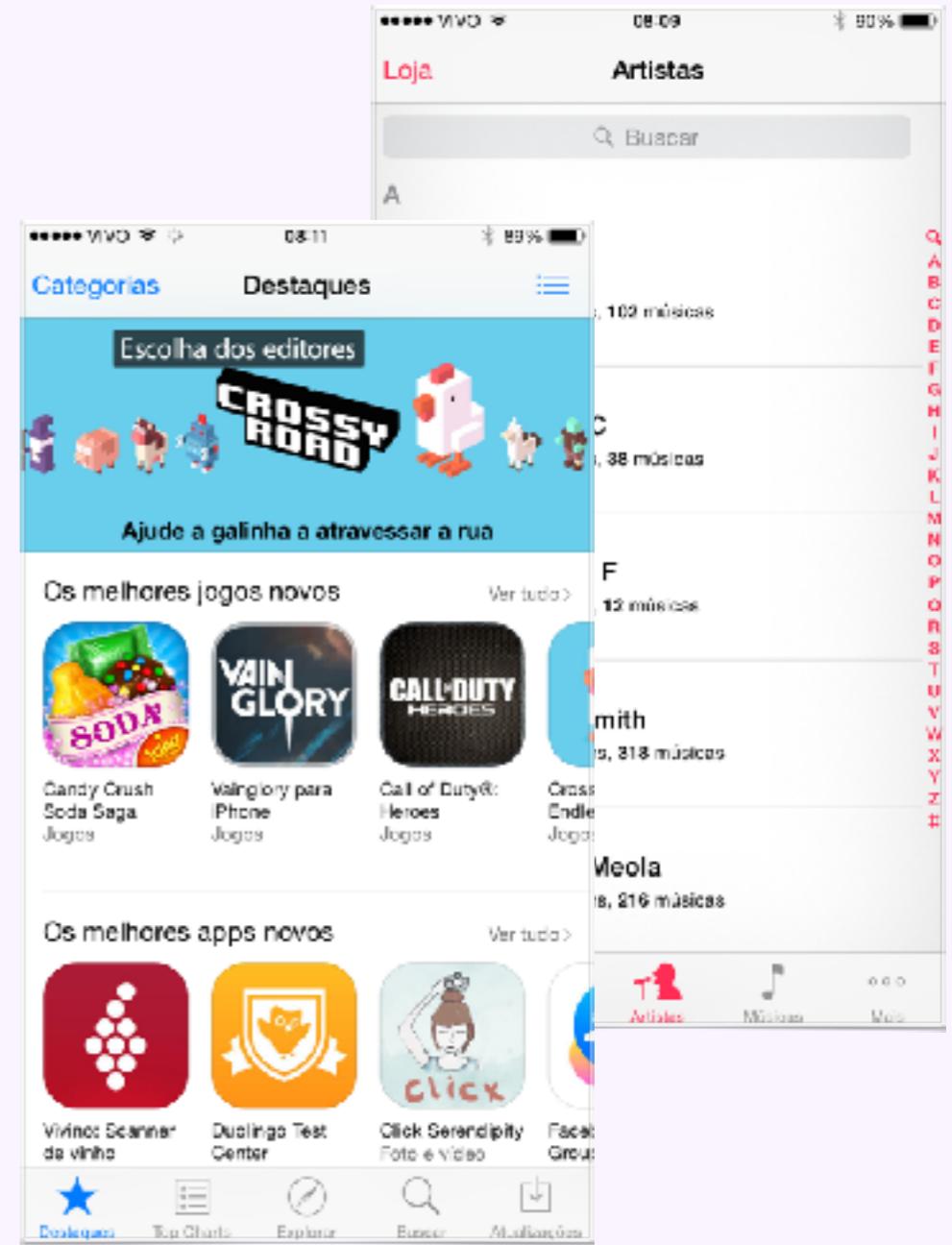
# TAB BAR CONTROLLER

# TAB BAR CONTROLLER

Permite construir interfaces com diversos segmentos ou seções.

Cada segmento é um *View Controller*.

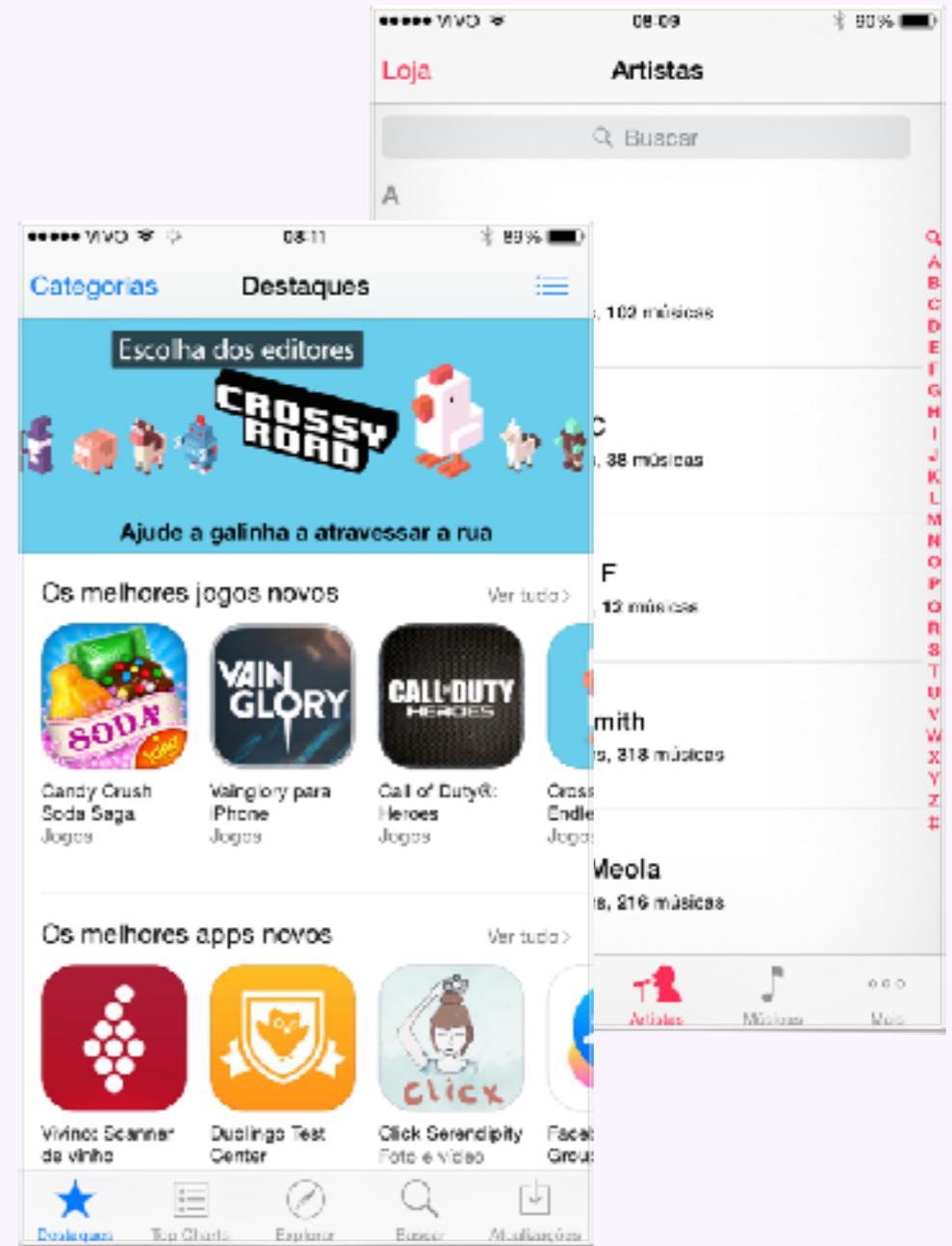
Possibilita ao usuário navegar entre as diversas seções do App através desse componente.

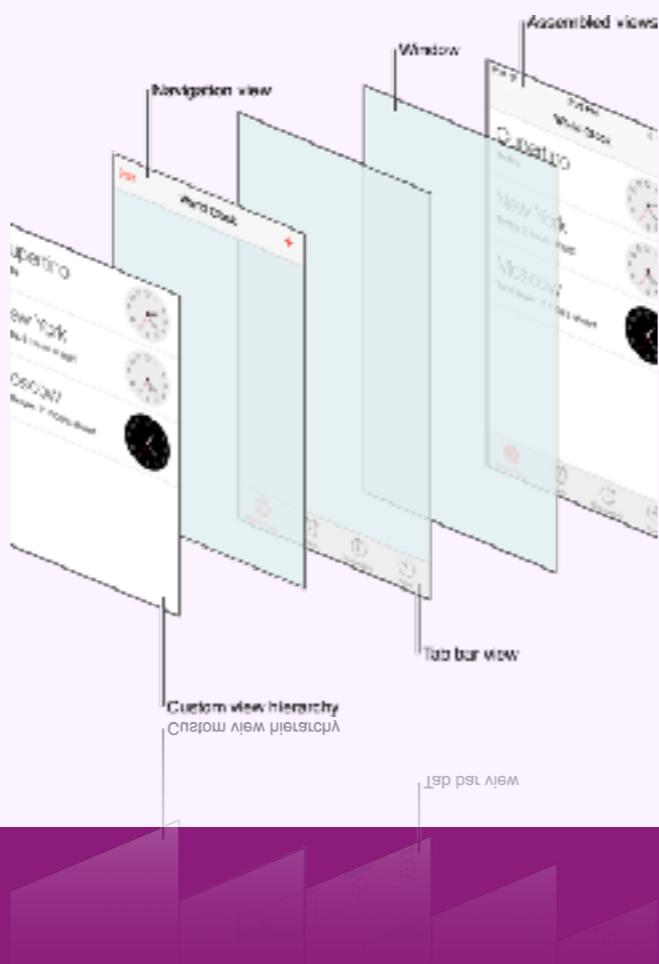


# TAB BAR CONTROLLER

Identifica visualmente as seções através de ícones e título

Exemplos de Apps: App Store, iTunes Store, Music, Fotos, Despertador



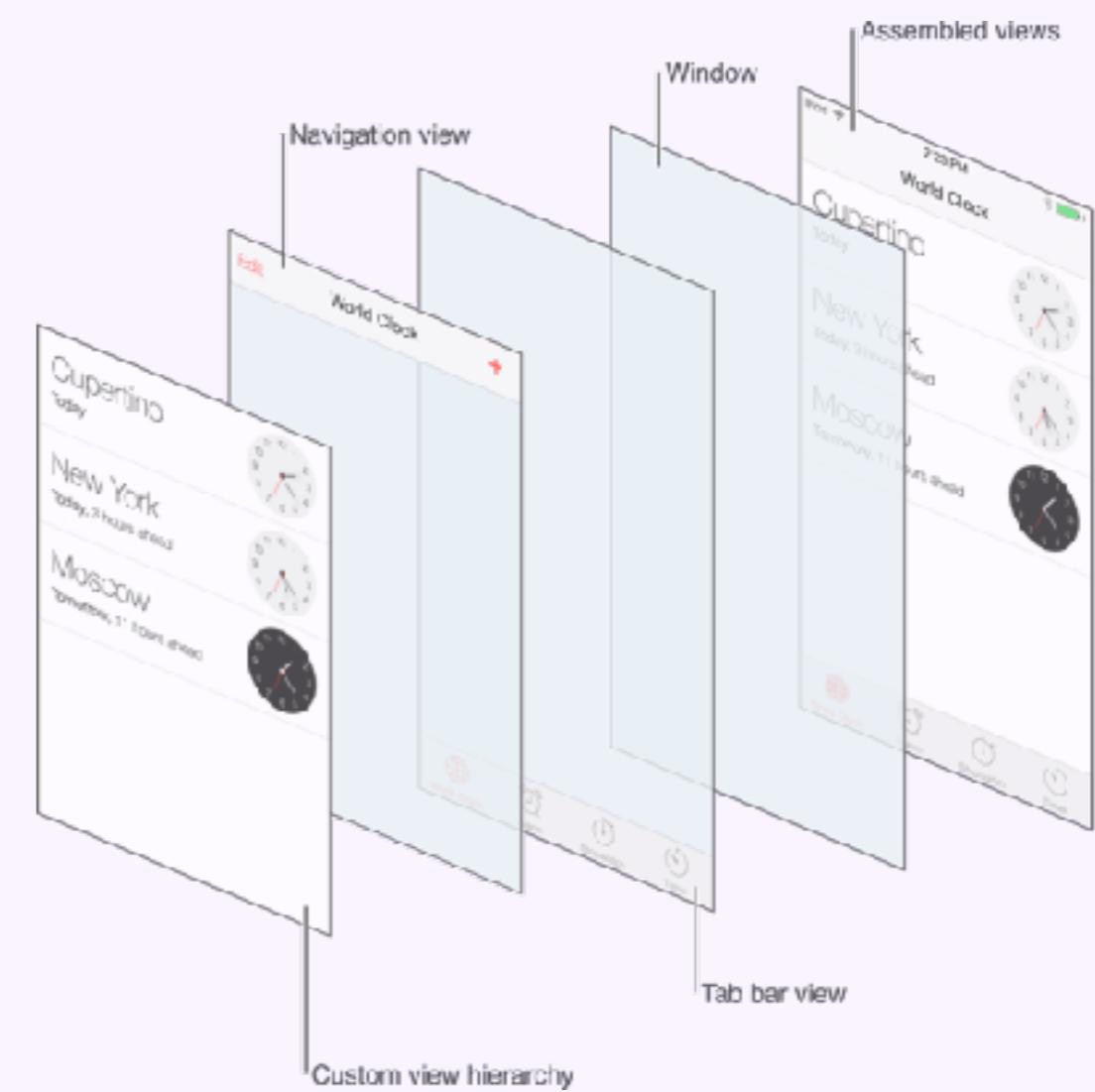


# VIEWS PERSONALIZADAS

# VIEWS PERSONALIZADAS

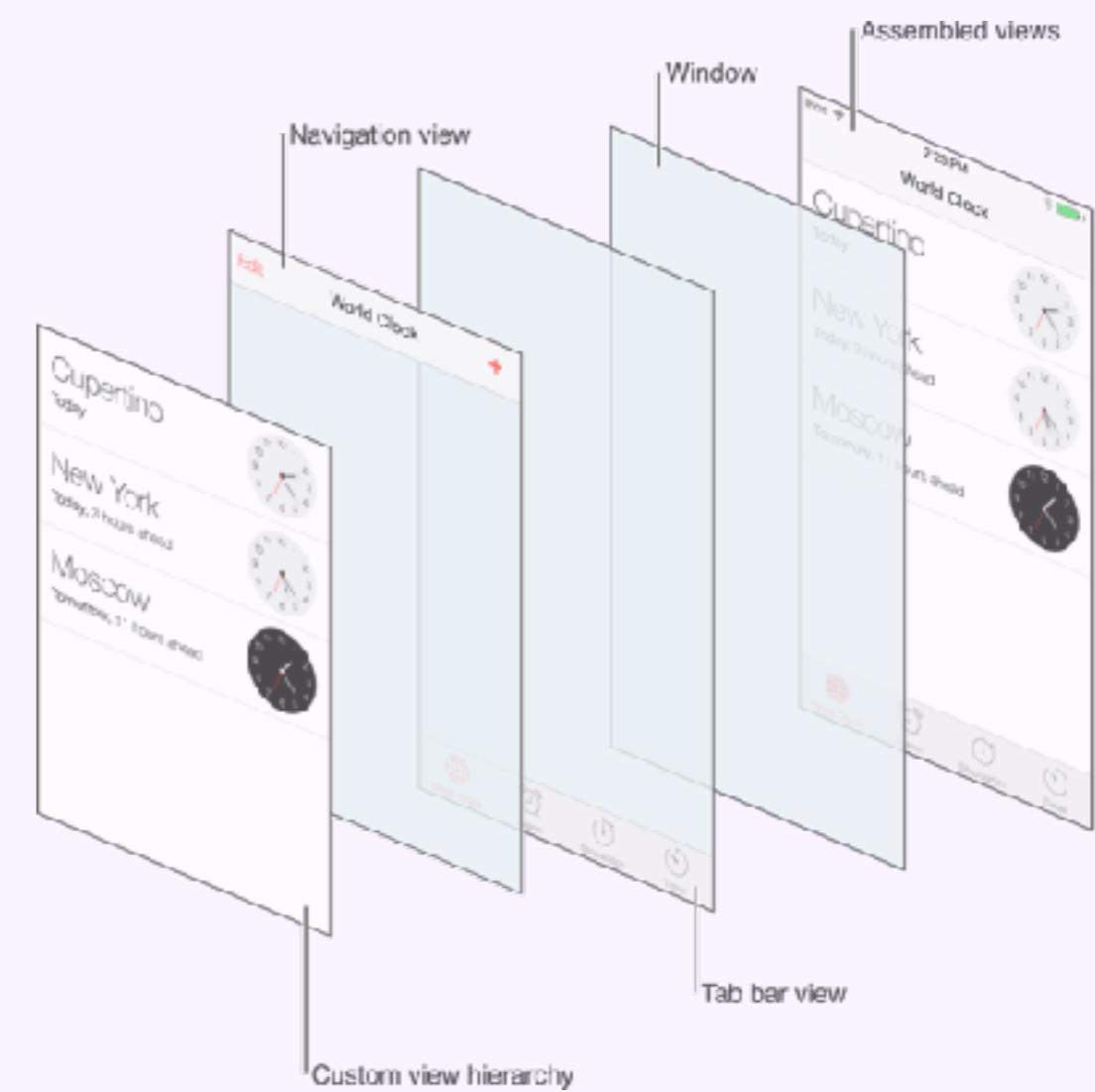
O objeto **UIView** pode ser estendido para criar Views e controles personalizados.

É um quadro onde formas primitivas podem ser desenhadas, ou outros objetos **UIView** inseridos, formando uma hierarquia de Views.



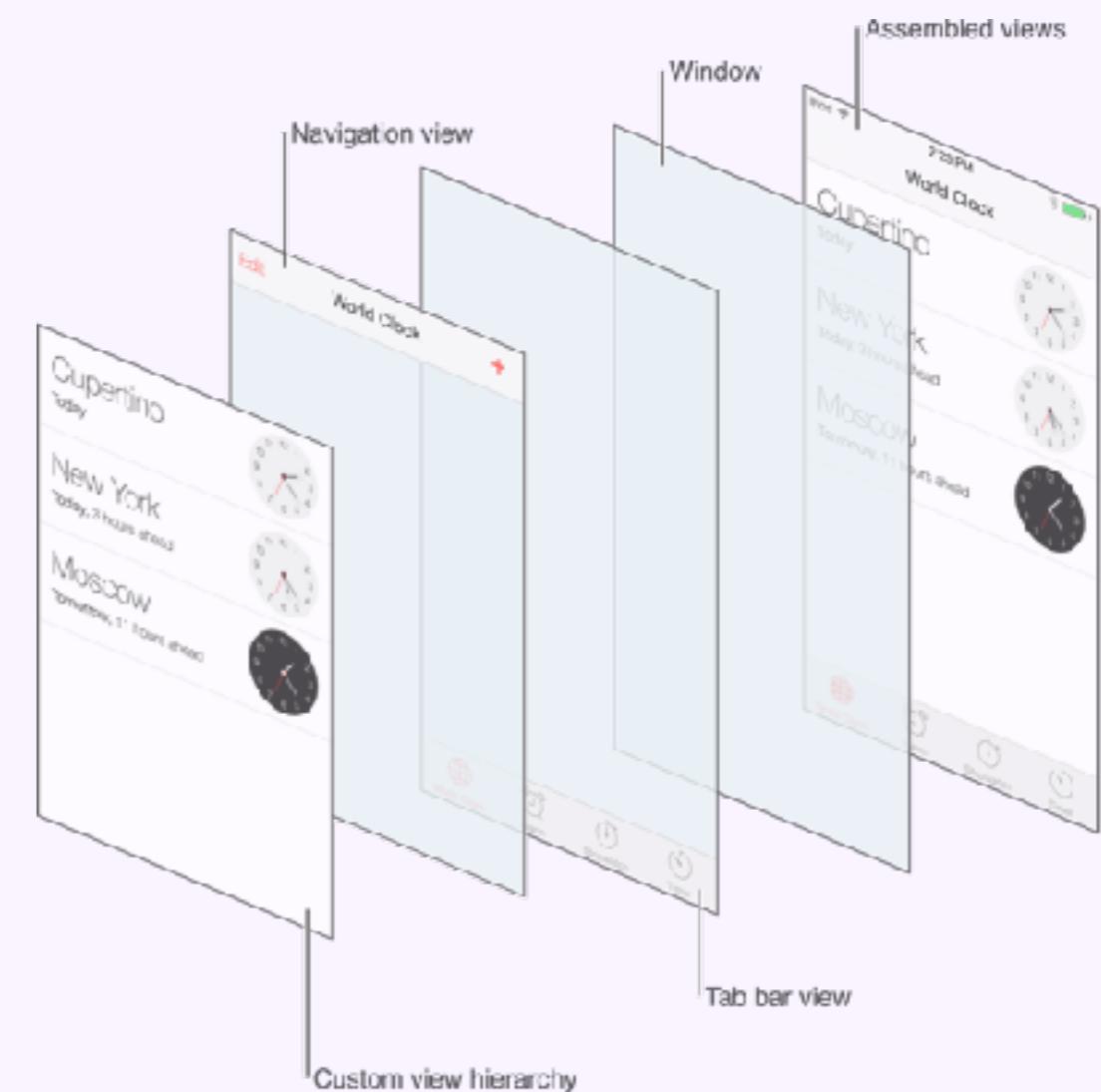
# VIEWS PERSONALIZADAS

São criados com classes que herdem de *UIView* e implementem seus métodos de desenho, ou compondo sua hierarquia de Views.



# VIEWS PERSONALIZADAS

Opcionalmente, podemos usar um Xib para desenhar os elementos de sua Interface, usando o mesmo mecanismo de *Outlets* e *Action Methods* para se comunicar com eles.





ANIMAÇÕES

# ANIMAÇÕES

O iOS inclui um Framework rico para criação de animações, o ***Core Animation***.

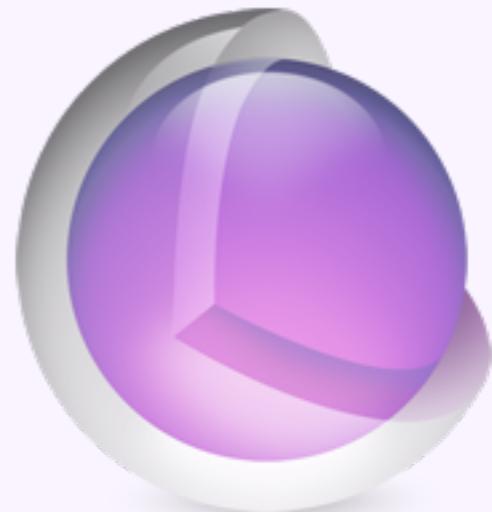
O **UIKit** integra o **Core Animation** para produzir animações em seus componentes.

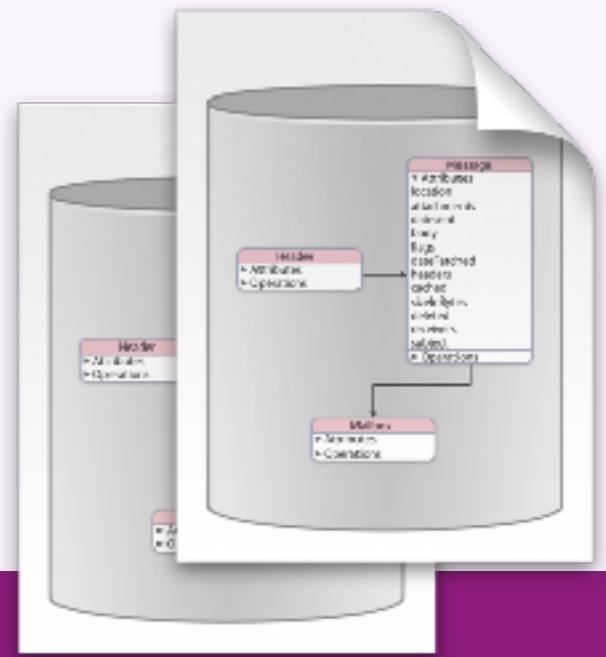


# ANIMAÇÕES

Podemos animar as propriedades do `UIView`: `frame`, `bounds`, `center`, `transform`, `alpha`, `backgroundColor` e `contentStretch`, entre outras.

Usando o `Auto-Layout`, `constraints` podem ser animadas, mais apropriado quando vamos movimentar o objeto pela tela.





# APRESENTANDO O CORE DATA

# APRESENTANDO O CORE DATA

Biblioteca de mapeamento objeto-relacional

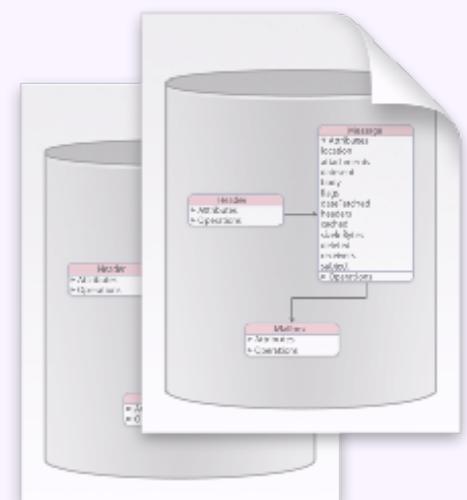
Funções:

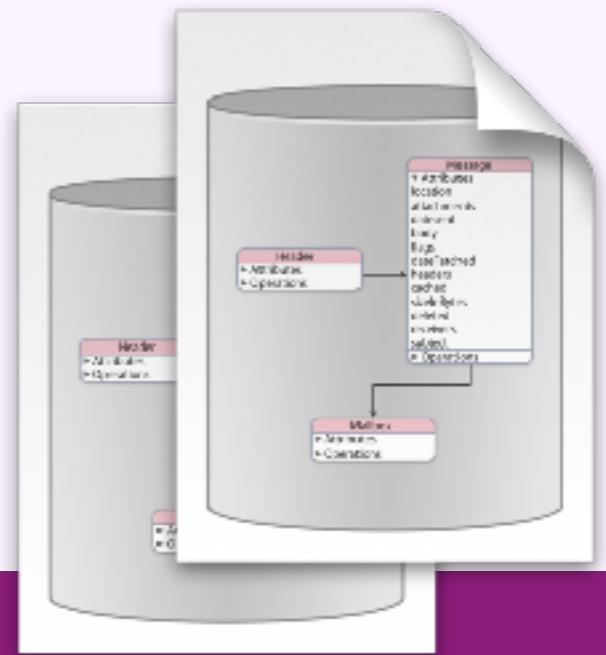
Construção de Modelos de Dados baseados em gráficos de Objetos

Persistência de Dados transparente com SQLite, XML (somente OSX) ou arquivos binários (somente OSX)

Mecanismo avançado de pesquisa de dados

Sincronização com iCloud



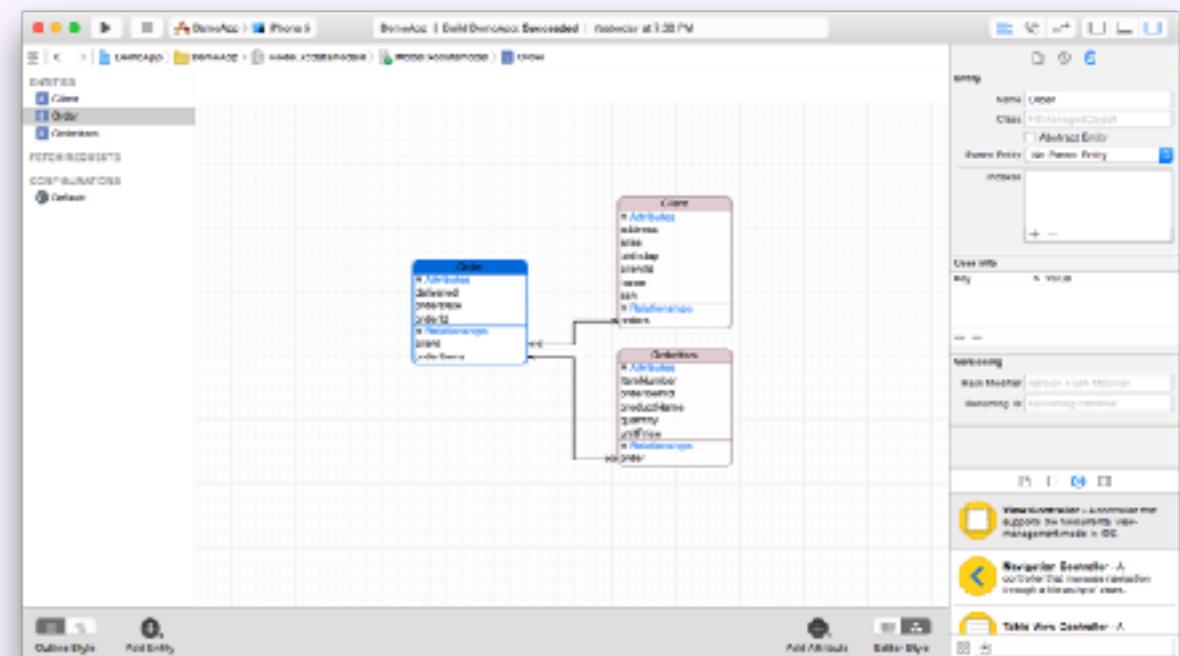


# MODELO DE DADOS

# MODELOS DE DADOS

Permitem construir gráficos de objetos a partir de entidades relacionadas.

Utiliza uma ferramenta gráfica para construção de Modelos.



# MODELO DO TRAQT

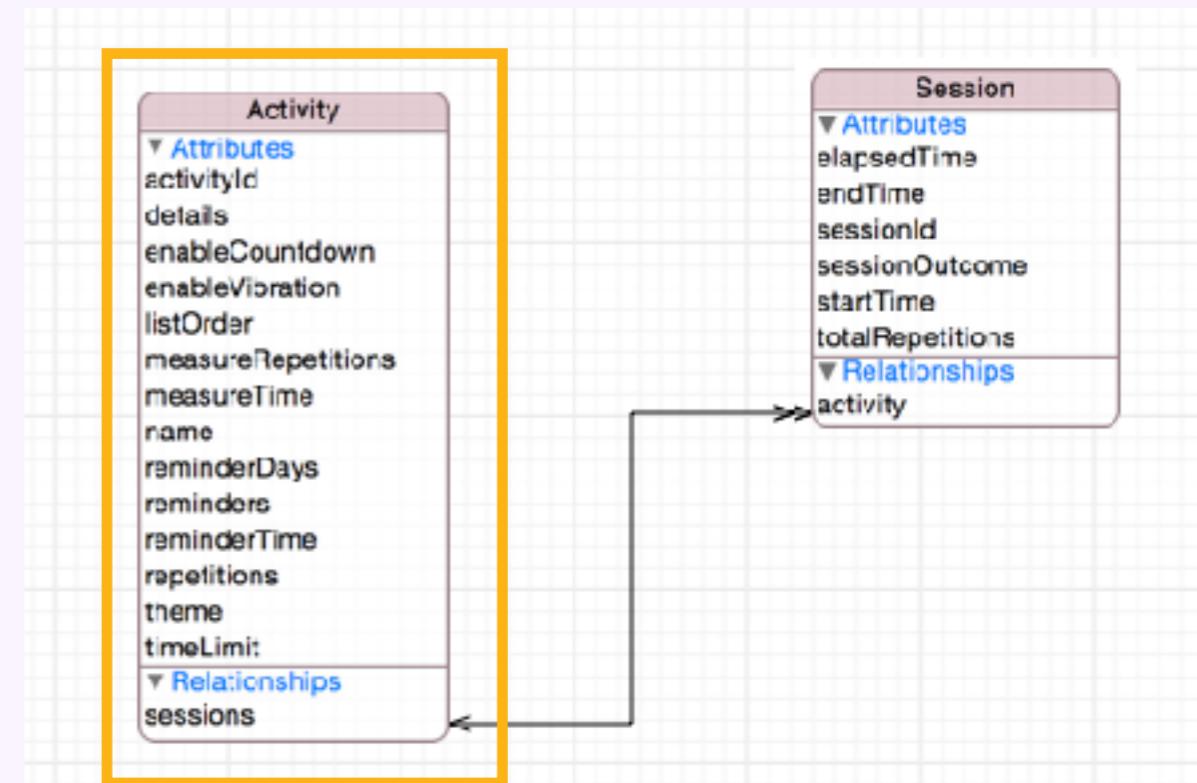
**Atividades:** registra as configurações de um tipo de atividade que o usuário deseja monitorar

Nome

Detalhes

Repetições

Tempo Limite



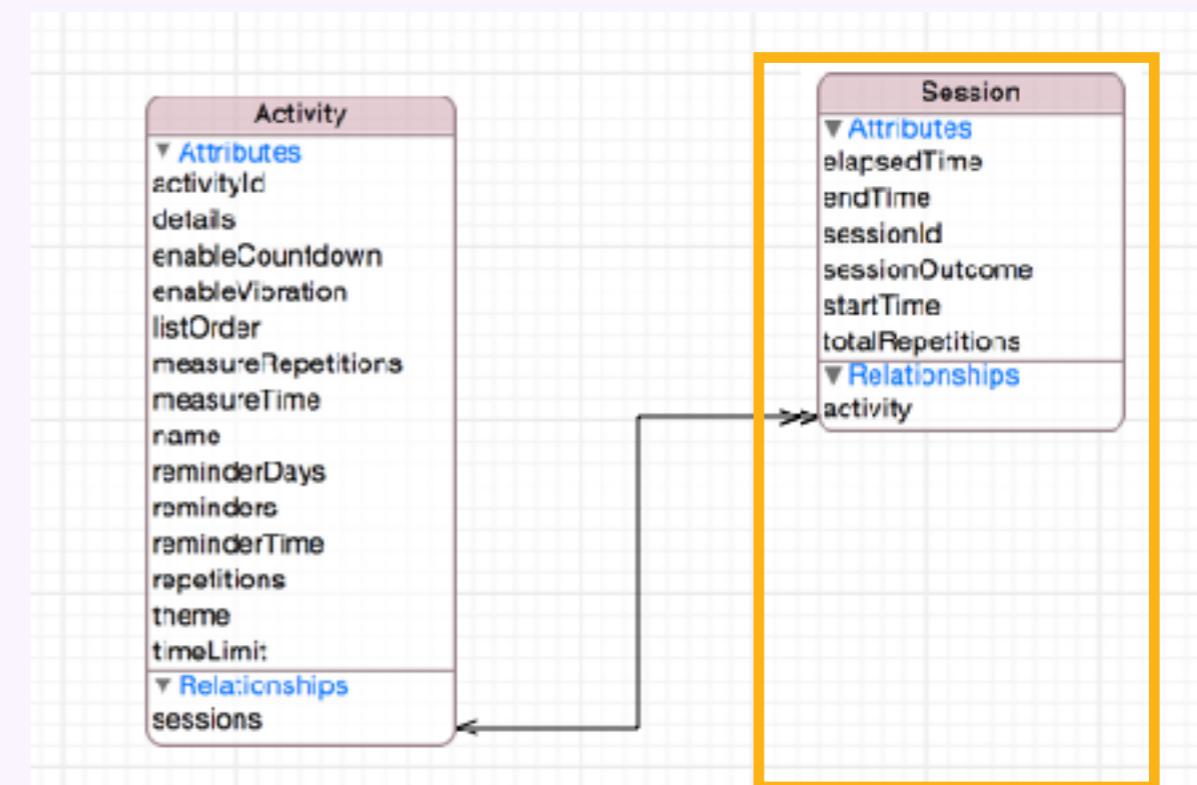
# MODELO DO TRAQT

**Sessões:** registrar as sessões de atividades executadas pelo usuário

Data da sessão

Tempo transcorrido

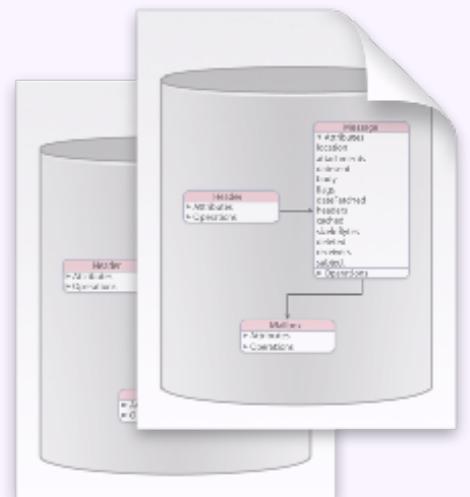
Total de Repetições



# A classe NSManagedObject

É a classe base para todos os objetos descritos como entidades do modelo.

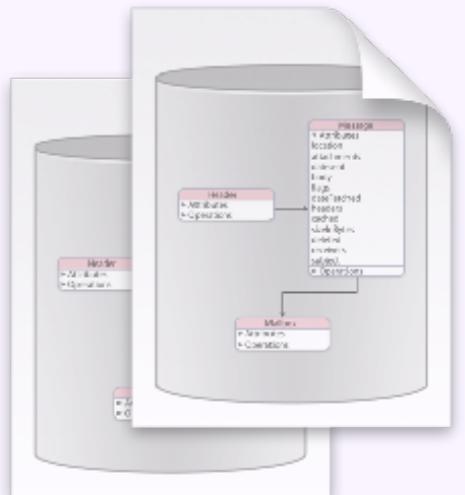
Proporciona as funcionalidades necessárias para manipular os dados da entidade, comunicar-se com o modelo, com as entidades relacionadas e acessar os eventos do *Core Data*.



# A classe NSManagedObject

Permite estender funcionalidades da entidade além dos atributos e relacionamentos definidos no modelo.

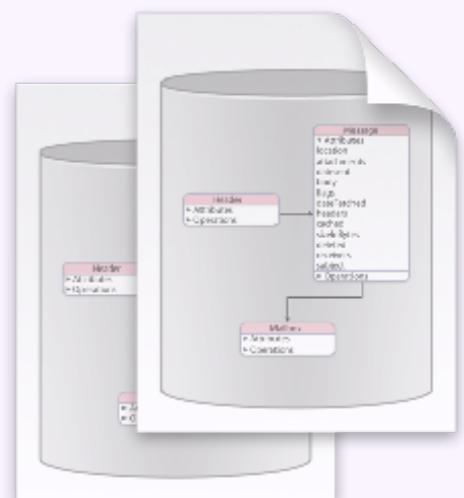
Devem ser exportadas a partir do *Editor de Modelo* do Core Data.



# A classe NSManagedObjectContext

Centraliza o acesso e a gestão dos registros das entidades.

Proporciona "caching" dos dados e monitora alterações feitas nas entidades carregadas em memória, garantindo sincronização com a camada de persistência.



# A biblioteca SwiftyIO

Componente criado pela Caeno, usando o *Swift* e os *Generics* para proporcionar acesso simplificado a um modelo do *Core Data*.

Facilita as operações de consulta, inclusão, edição e exclusão de dados das entidades.

Construída sobre as funcionalidades do  
**NSManagedObjectContext**.

