

cif



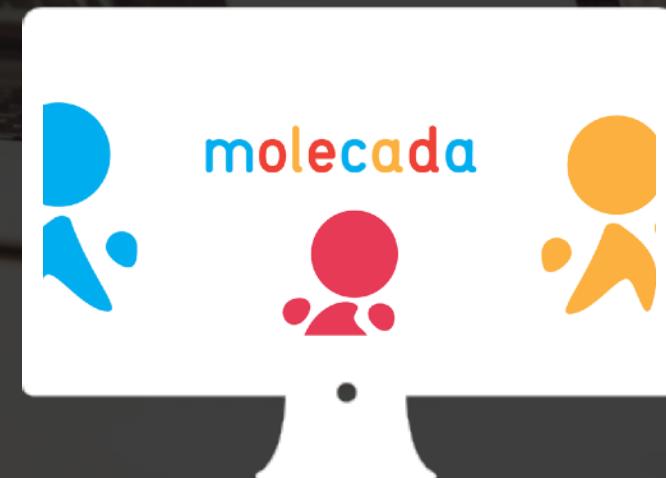
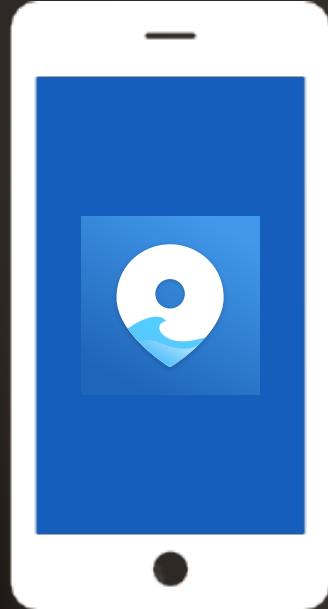
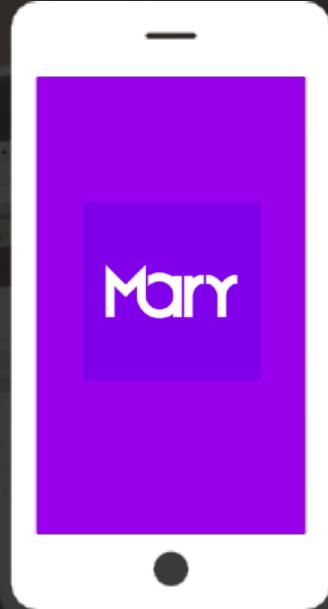
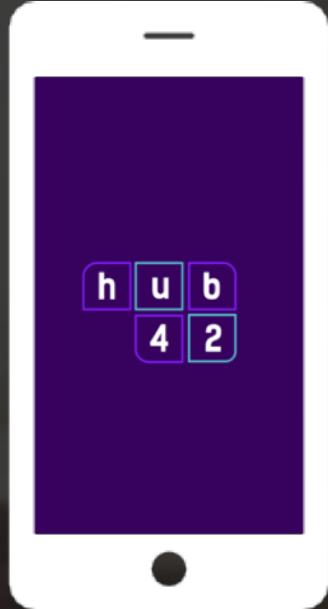
Hilton Pintor

Desenvolvedor (iOS/tvOS/watchOS)

hiltonpintor@gmail.com



// Aula 01



citi

Icons made by Freepik and Dale Humphries from flaticon.com

iOS
com Swift



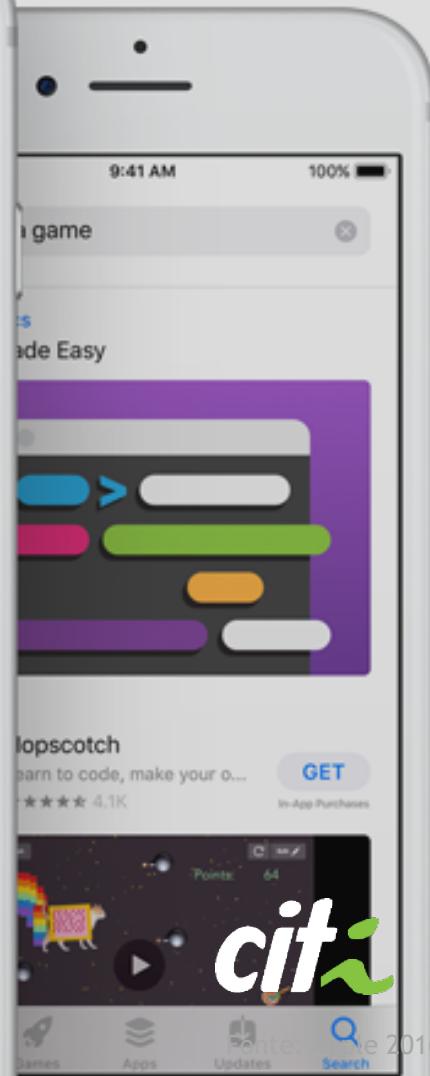
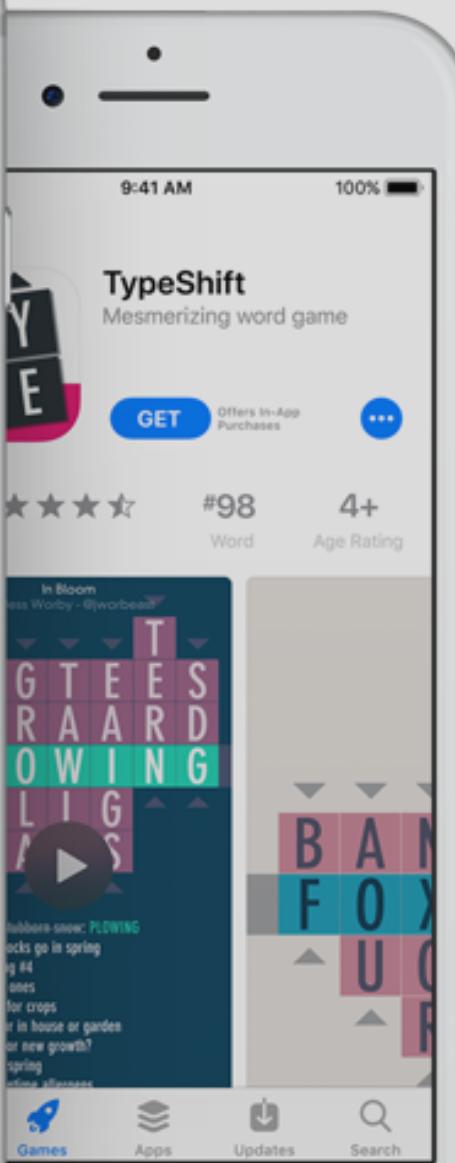
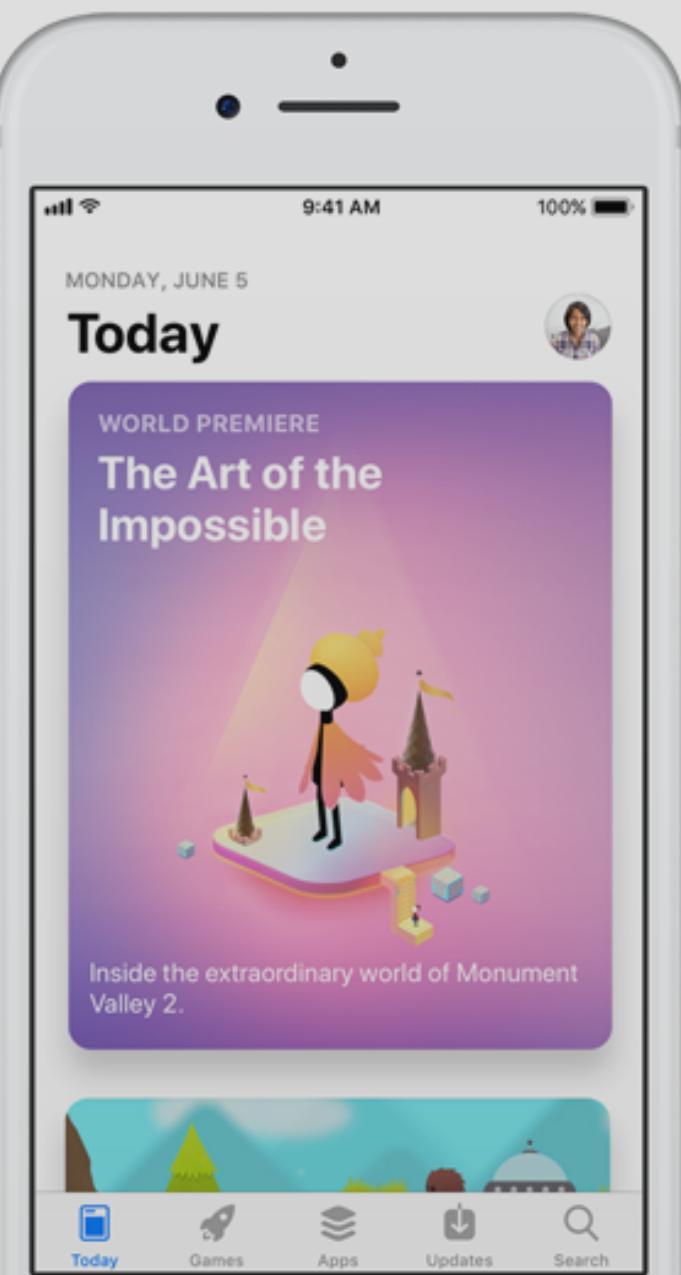
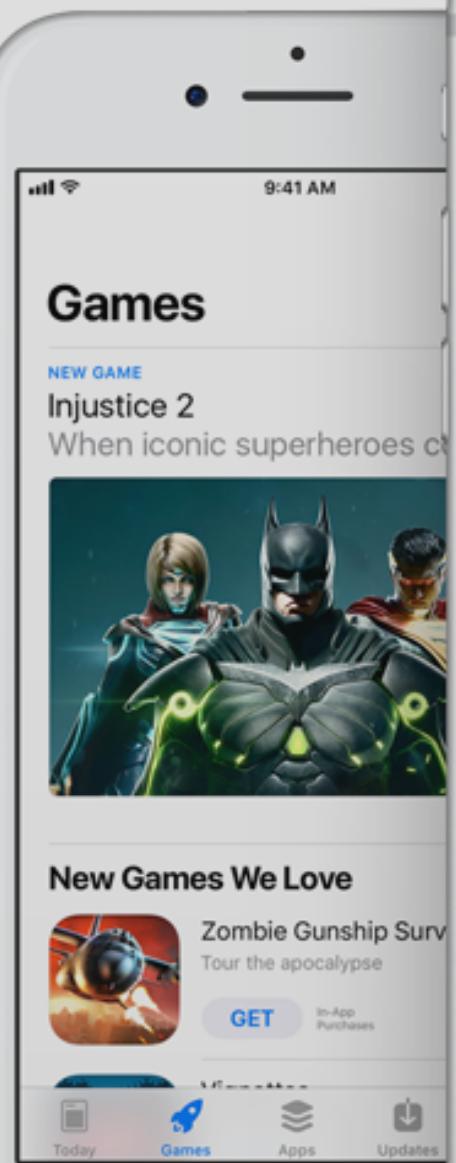
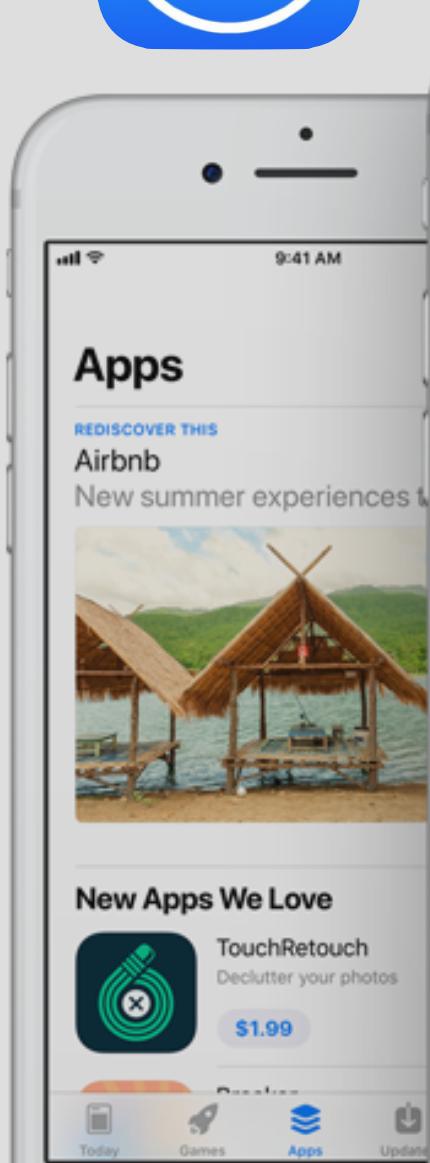
- Nome
- Ocupação
- Experiência com programação
- Expectativa com o curso

iOS
com Swift





App Store





2
milhões
apps

em 2016

218 bi

Reais

desde 2008



citi

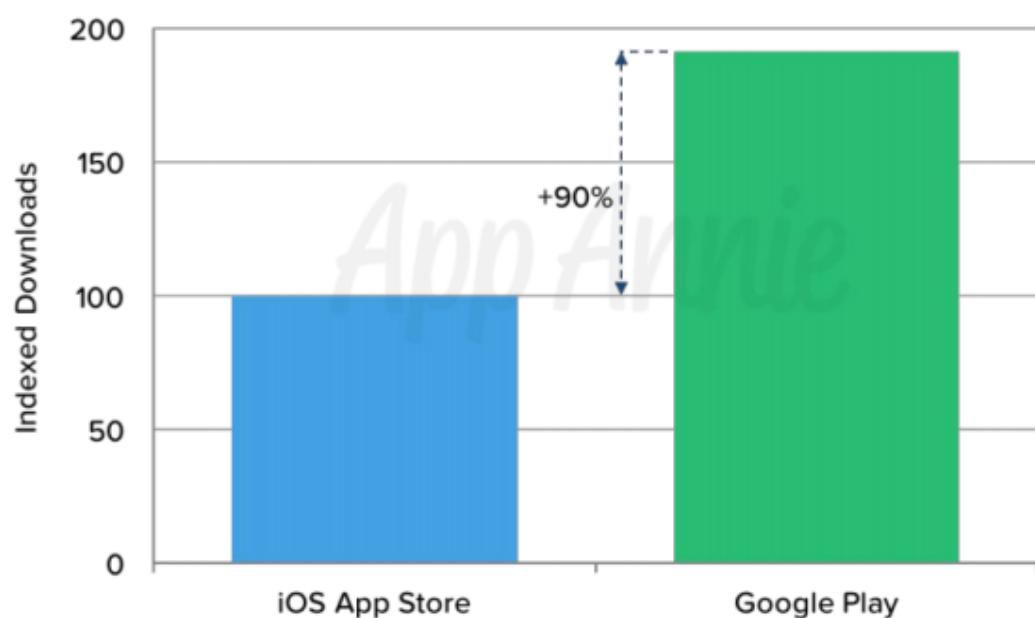
Fonte: Apple 2017



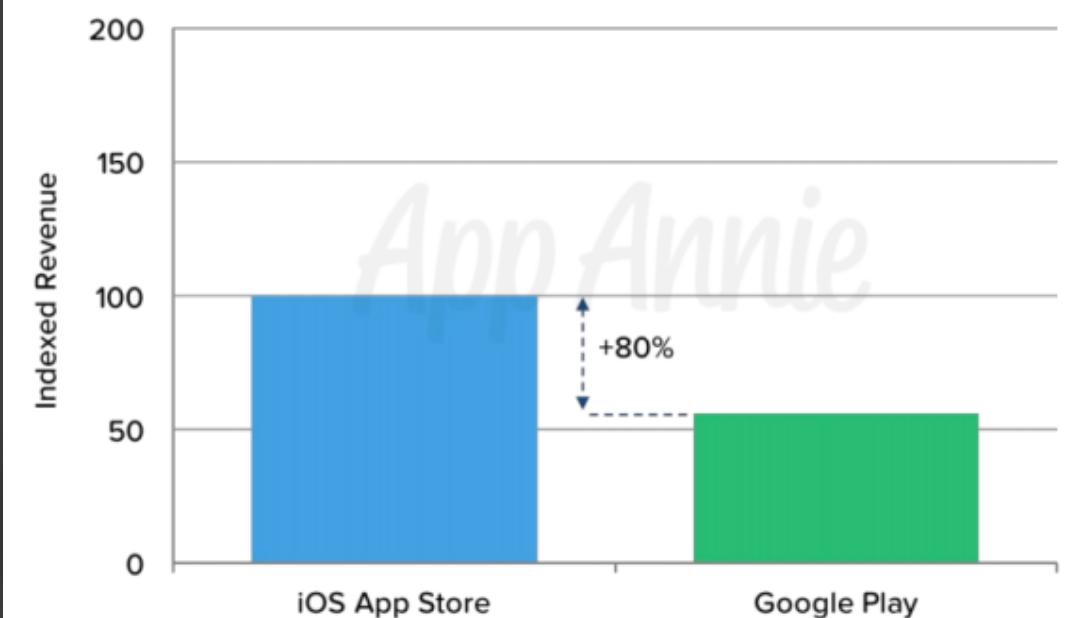
citi

Fonte: Apple 2017

Worldwide App Downloads by Store
Q3 2015



Worldwide App Revenue by Store
Q3 2015



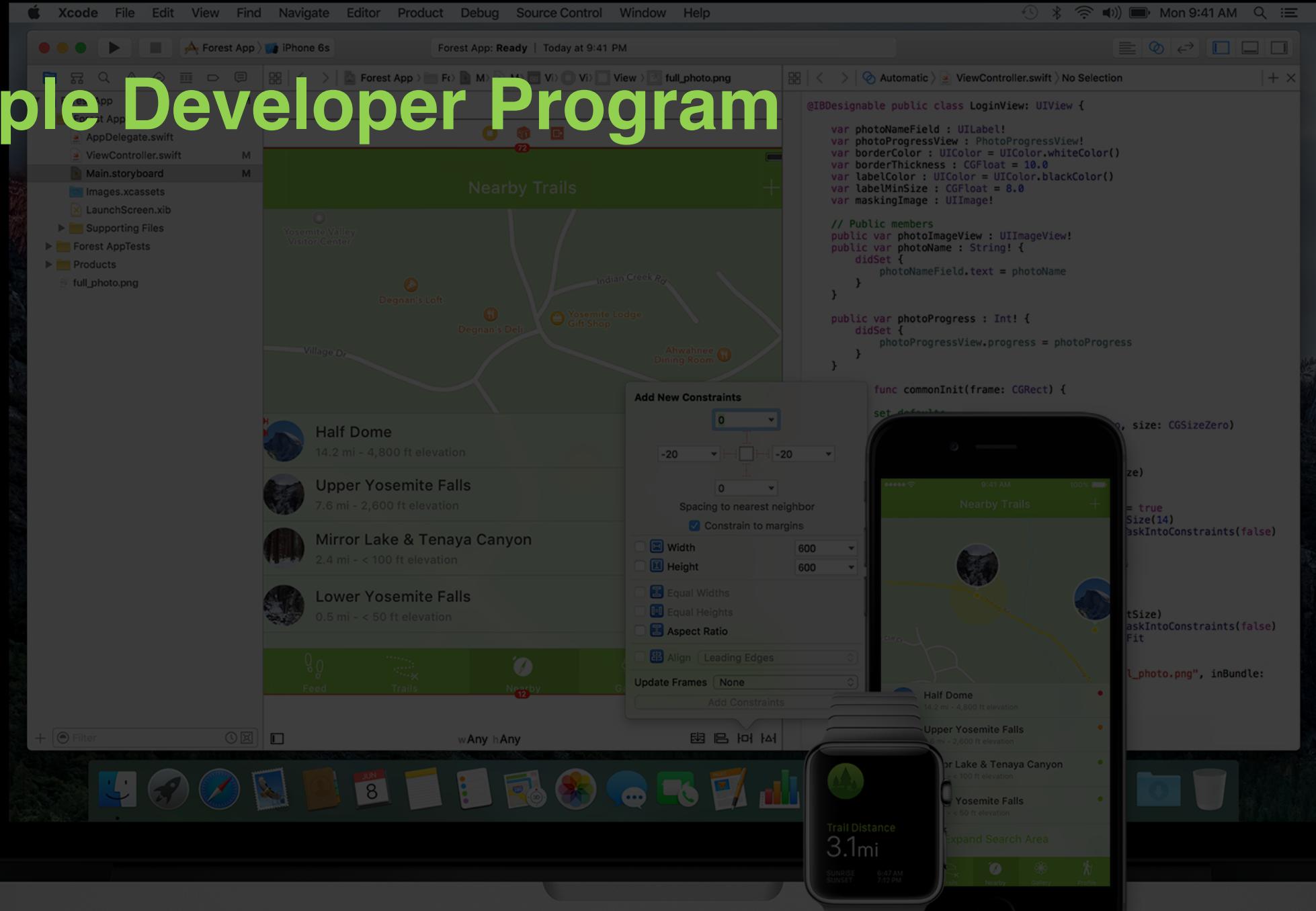
Installed base



citi

Fonte: Apple 2017

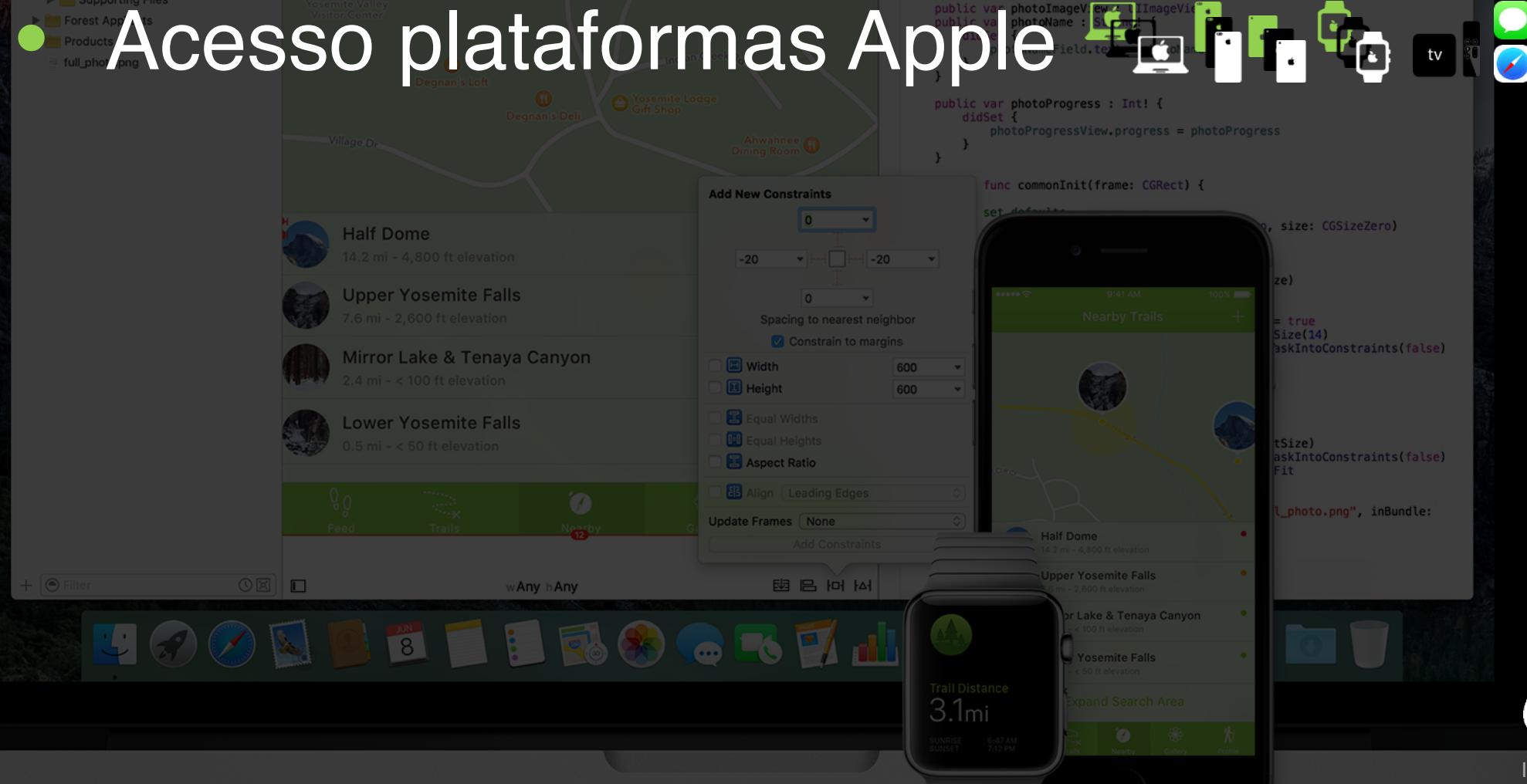
Apple Developer Program



citi

Image by Apple

Apple Developer Program



citi

Image by Apple

Apple Developer Program



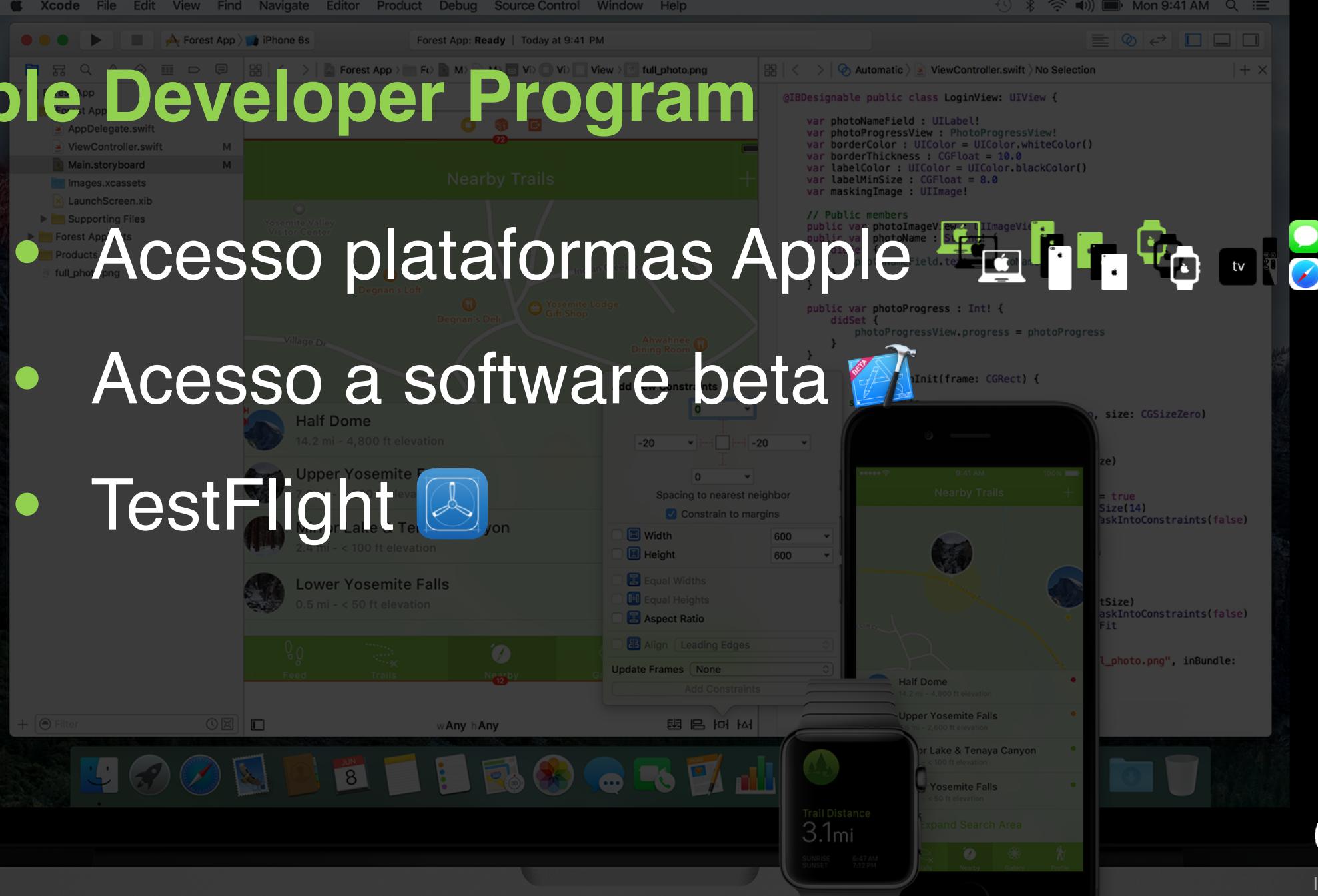
- Acesso plataformas Apple
- Acesso a software beta

citi

Image by Apple

Apple Developer Program

- Acesso plataformas Apple
- Acesso a software beta
- TestFlight 

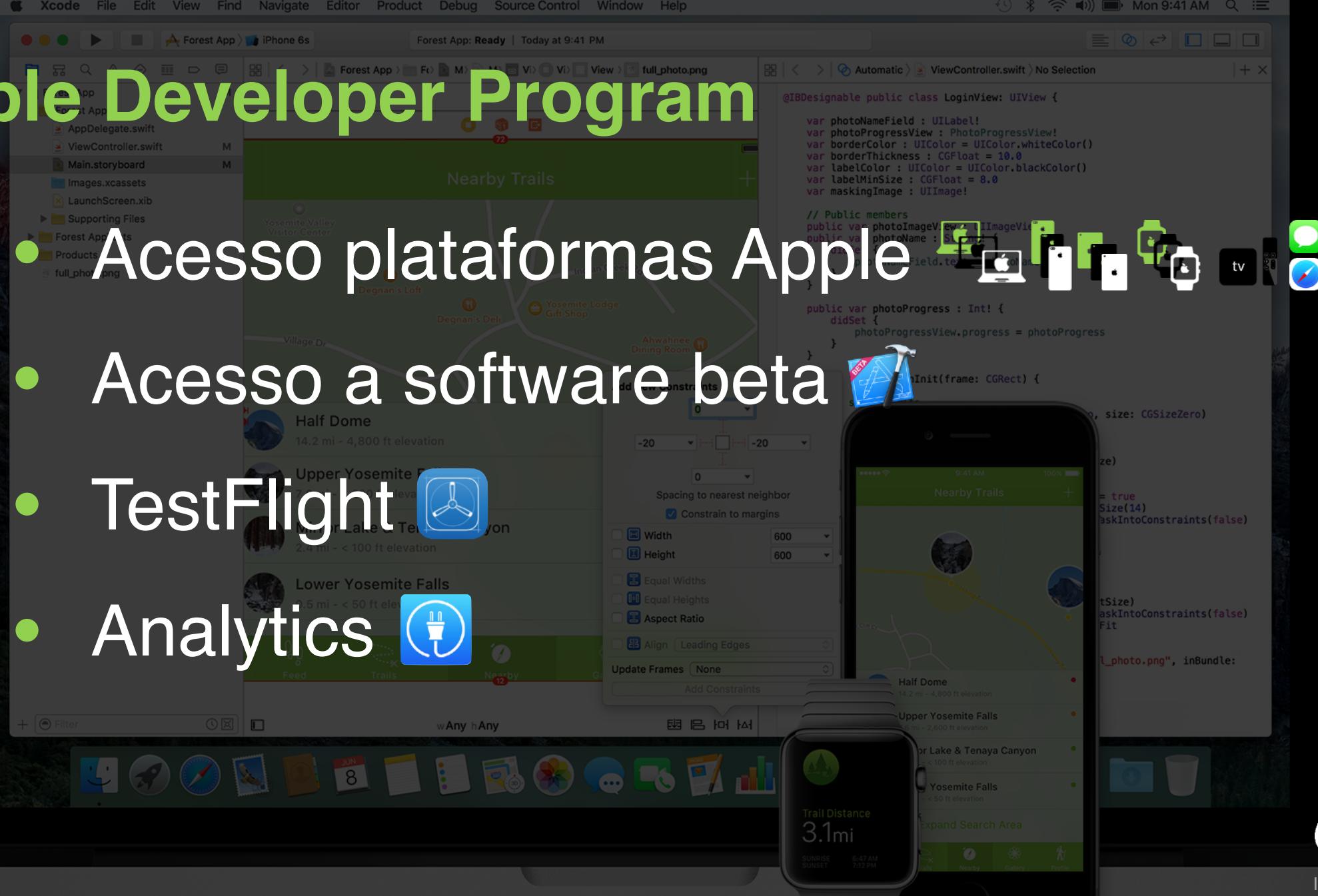


The screenshot shows the Xcode IDE. In the center, there's a preview of an iPhone displaying a map application titled "Nearby Trails". The map shows various trails in Yosemite National Park, including Half Dome, Upper Yosemite Falls, and Lower Yosemite Falls. On the right side of the screen, the "ViewController.swift" file is open, showing Swift code for a "LoginView" class. The code includes properties like "photoNameField", "photoProgressView", and "borderColor". It also contains methods for setting photo progress and initializing the view. At the bottom right of the slide, there's a "citi" logo.

citi

Image by Apple

Apple Developer Program

- Acesso plataformas Apple
 - Acesso a software beta
 - TestFlight 
 - Analytics 
- 
- The background image shows a screenshot of the Xcode IDE. On the left, the Project Navigator displays files like AppDelegate.swift, ViewController.swift, Main.storyboard, and LaunchScreen.xib. The main canvas shows a map application titled 'Nearby Trails' with locations like 'Yosemite Valley Visitor Center', 'Degnan's Loft', 'Yosemite Lodge Gift Shop', 'Ahwahnee Dining Room', 'Half Dome', 'Upper Yosemite Falls', and 'Lower Yosemite Falls'. The bottom right corner of the screen shows a watch face with the text 'Trail Distance 3.1mi'. On the right side of the Xcode interface, the code editor shows Swift code for a 'LoginView' class, including properties for a photo name field, progress view, border color, and thickness. There are also sections for public members and a 'didSet' block for the photo progress variable. The status bar at the top right indicates it's Monday at 9:41 AM.

citi

Image by Apple

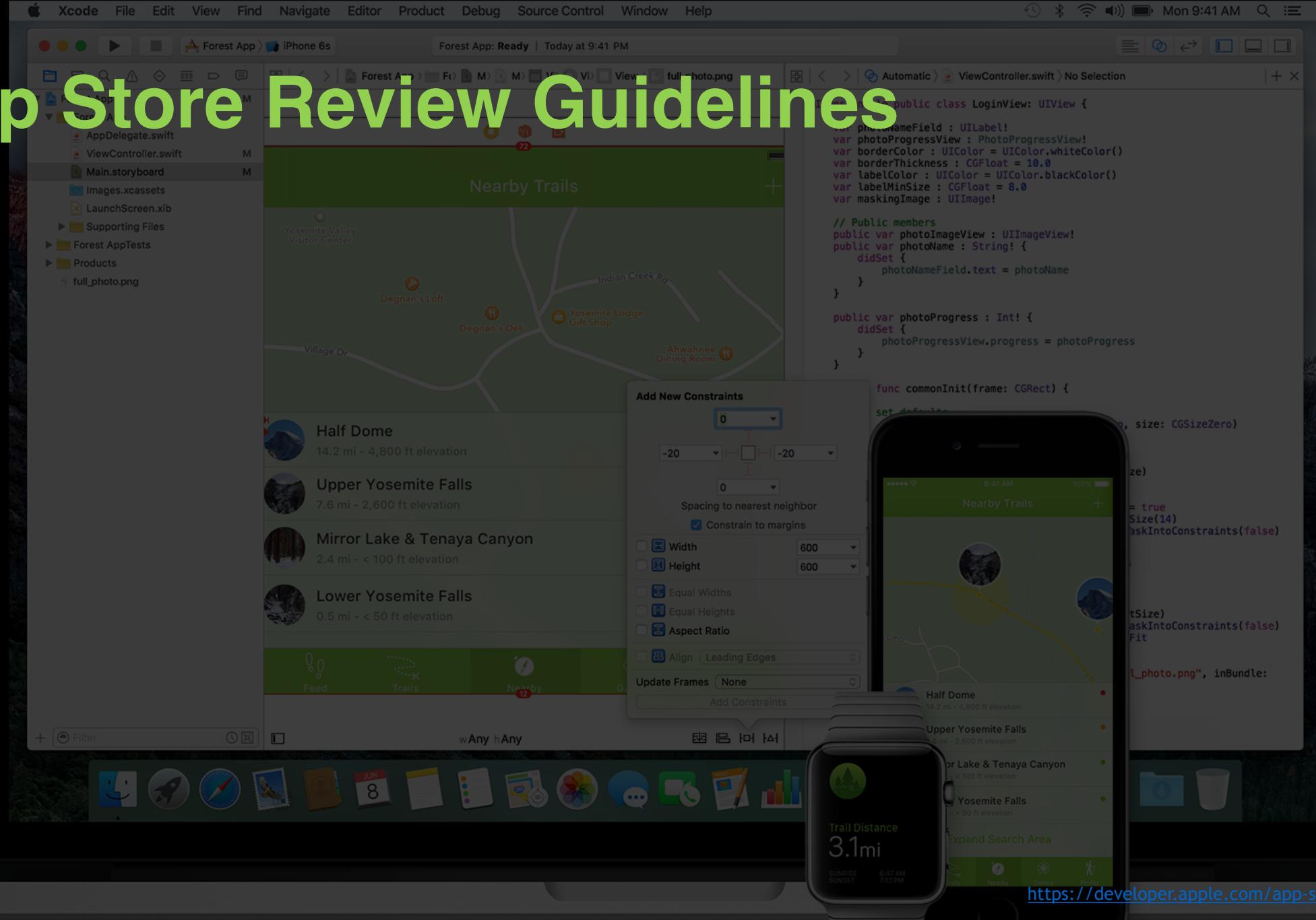
Apple Developer Program

- Acesso plataformas Apple
 - Acesso a software beta
 - TestFlight 
 - Analytics 
 - 99 USD 
- 
- The background image shows a screenshot of the Xcode IDE. On the left, the Project Navigator displays files like AppDelegate.swift, ViewController.swift, Main.storyboard, and LaunchScreen.xib. The main canvas shows a map application titled 'Nearby Trails' with locations like 'Yosemite Valley Visitor Center', 'Degnan's Loft', 'Yosemite Lodge Gift Shop', and 'Ahwahnee Dining Room'. On the right, the Assistant Editor shows a Swift file named ViewController.swift with code related to 'LoginView' and 'PhotoProgressView'. Below the code editor, there are icons for various Apple devices: iPhone, iPad, Mac, Watch, and TV. At the bottom right, there is a 'citi' logo.

citi

Image by Apple

App Store Review Guidelines

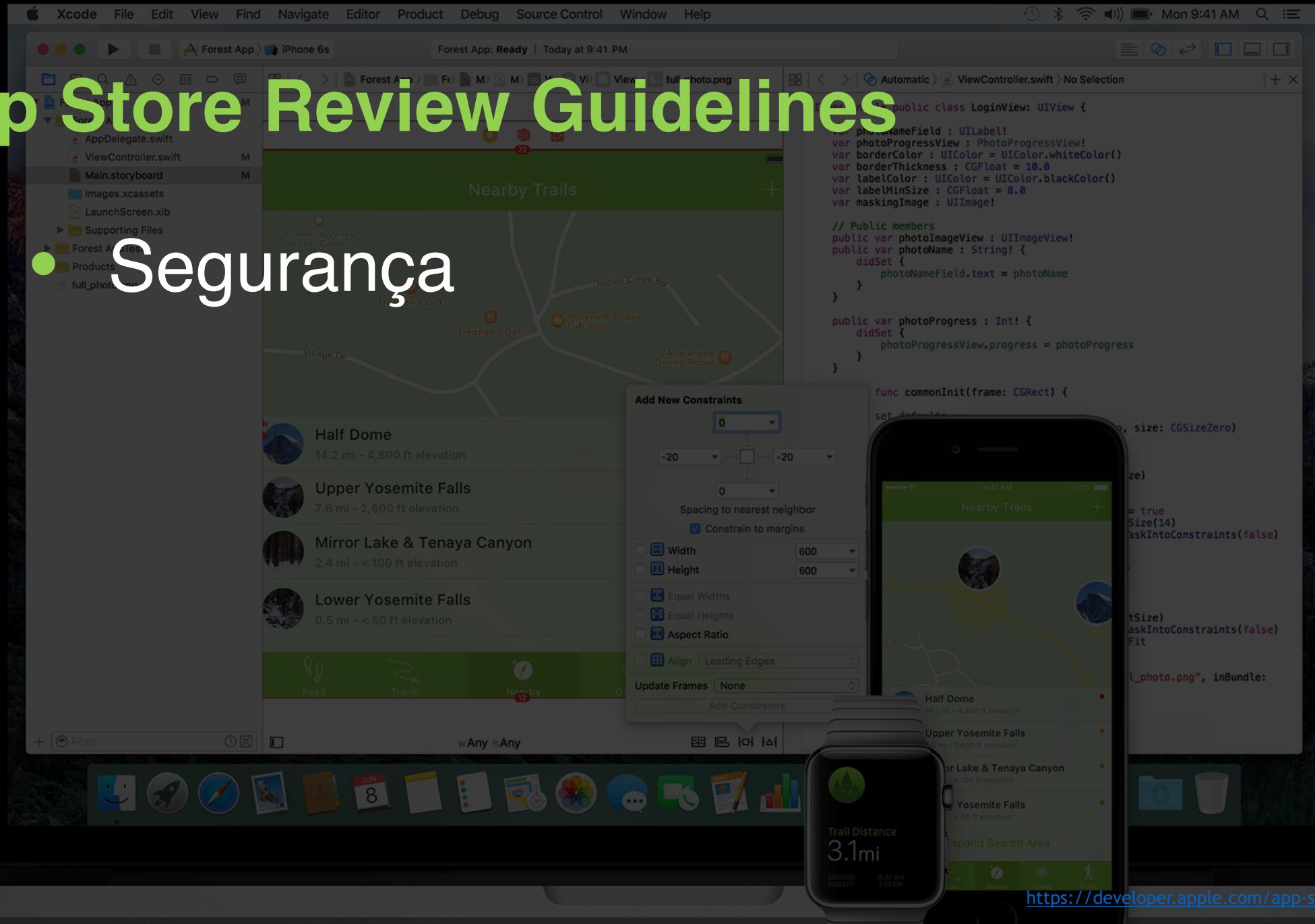


citi

<https://developer.apple.com/app-store/review/guidelines/>

App Store Review Guidelines

Segurança

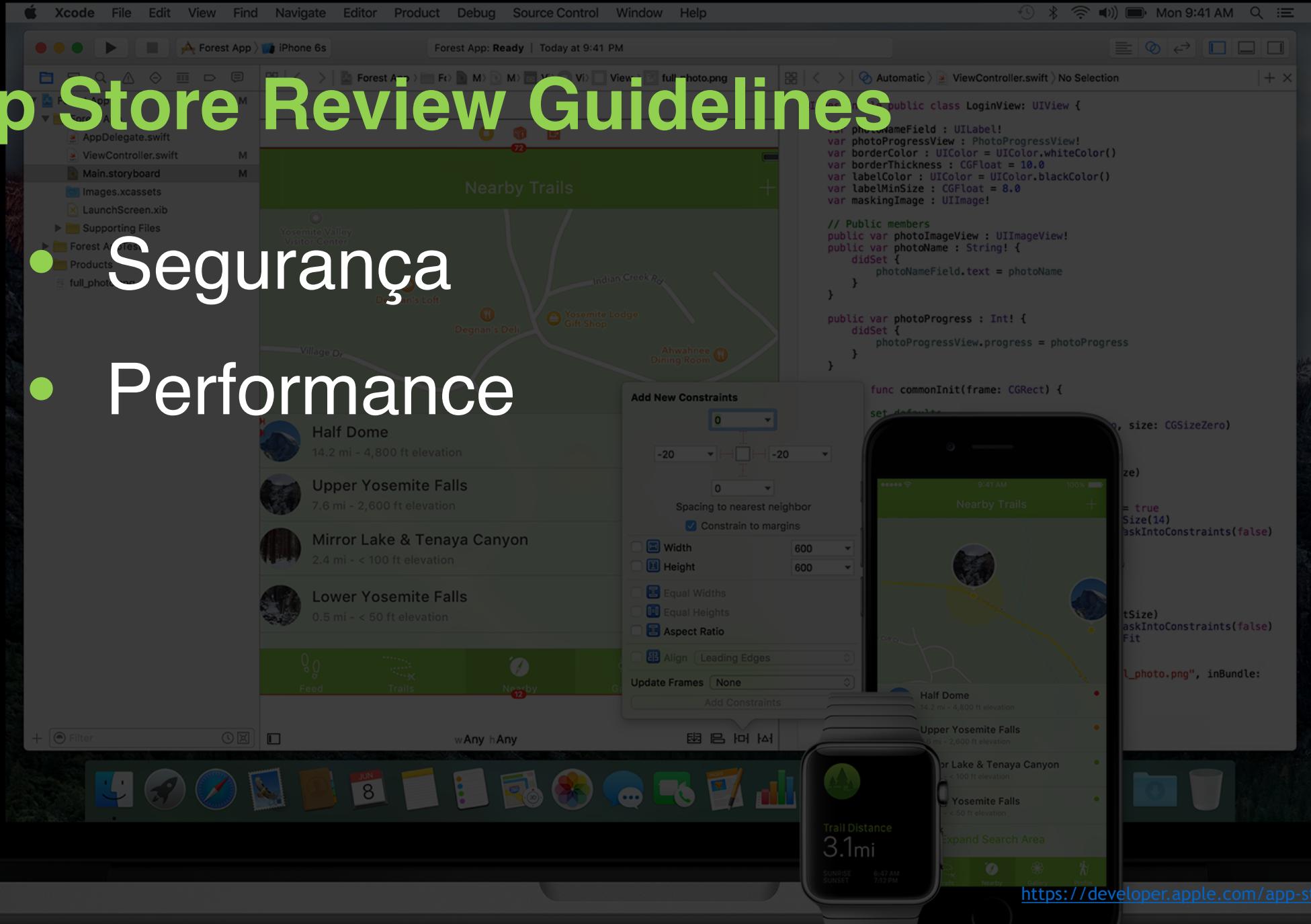


citi

<https://developer.apple.com/app-store/review/guidelines/>

App Store Review Guidelines

- Segurança
- Performance



citi

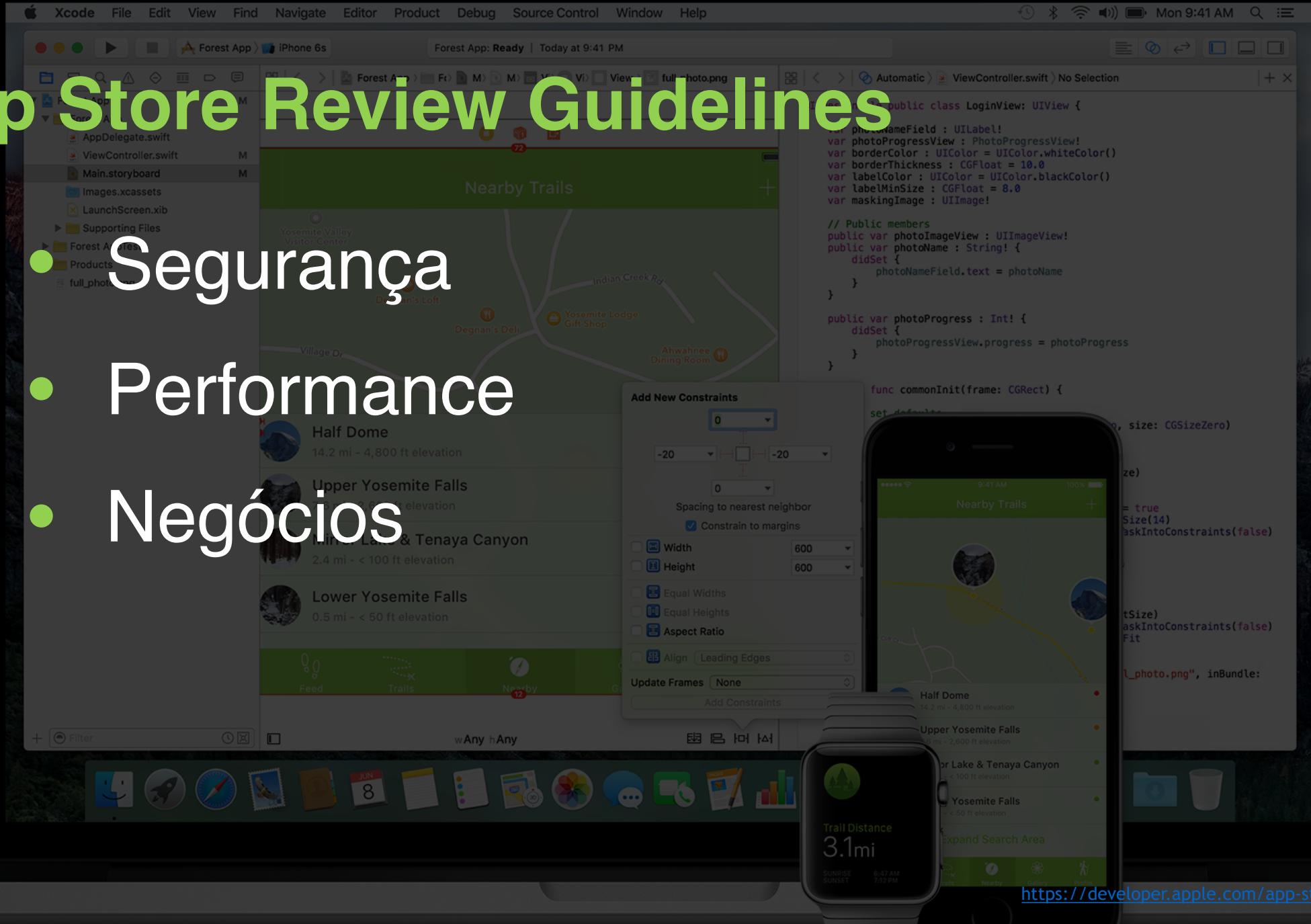
<https://developer.apple.com/app-store/review/guidelines/>

App Store Review Guidelines

- Segurança

- Performance

- Negócios



citi

<https://developer.apple.com/app-store/review/guidelines/>

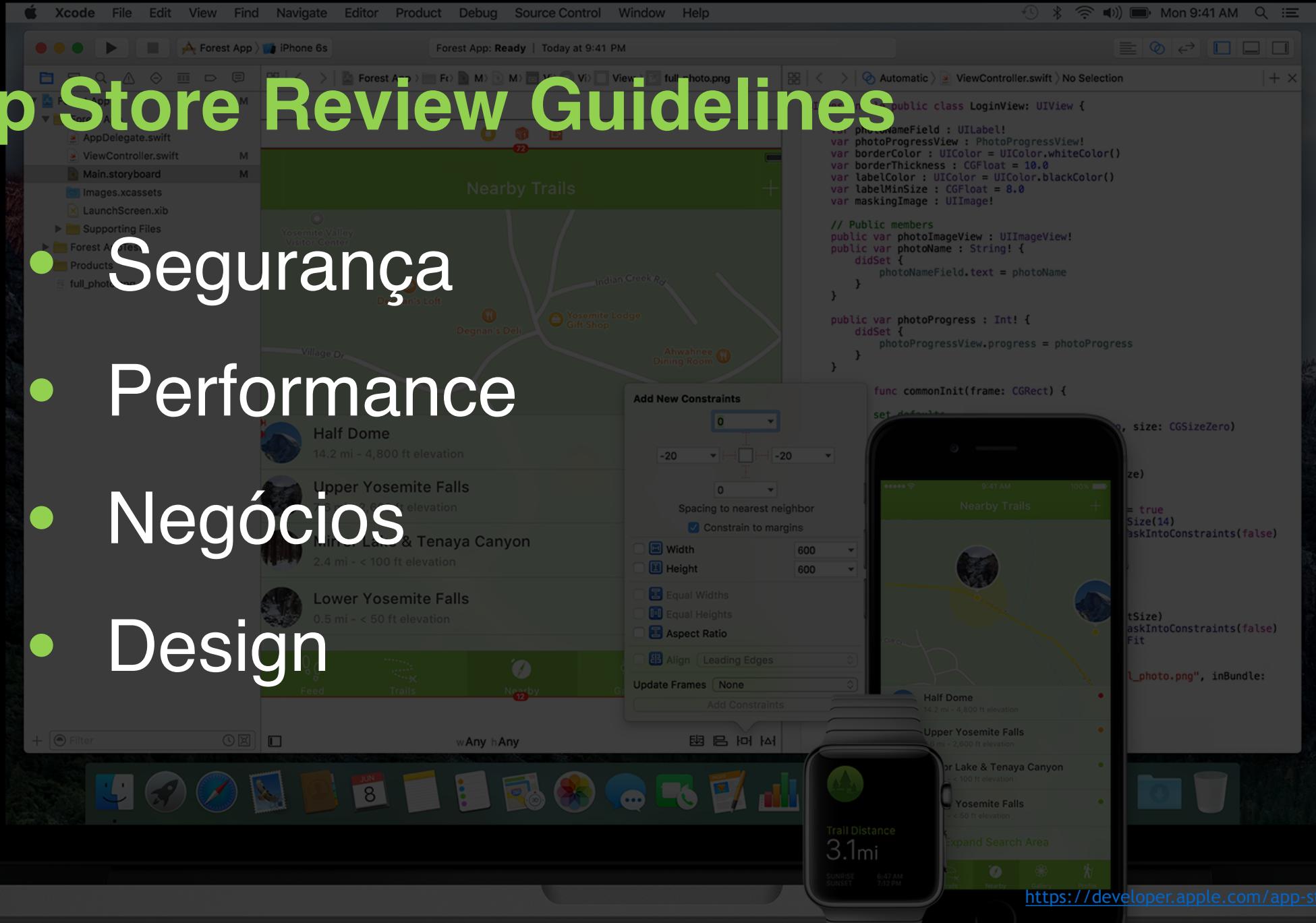
App Store Review Guidelines

- Segurança

- Performance

- Negócios

- Design



citi

<https://developer.apple.com/app-store/review/guidelines/>

App Store Review Guidelines

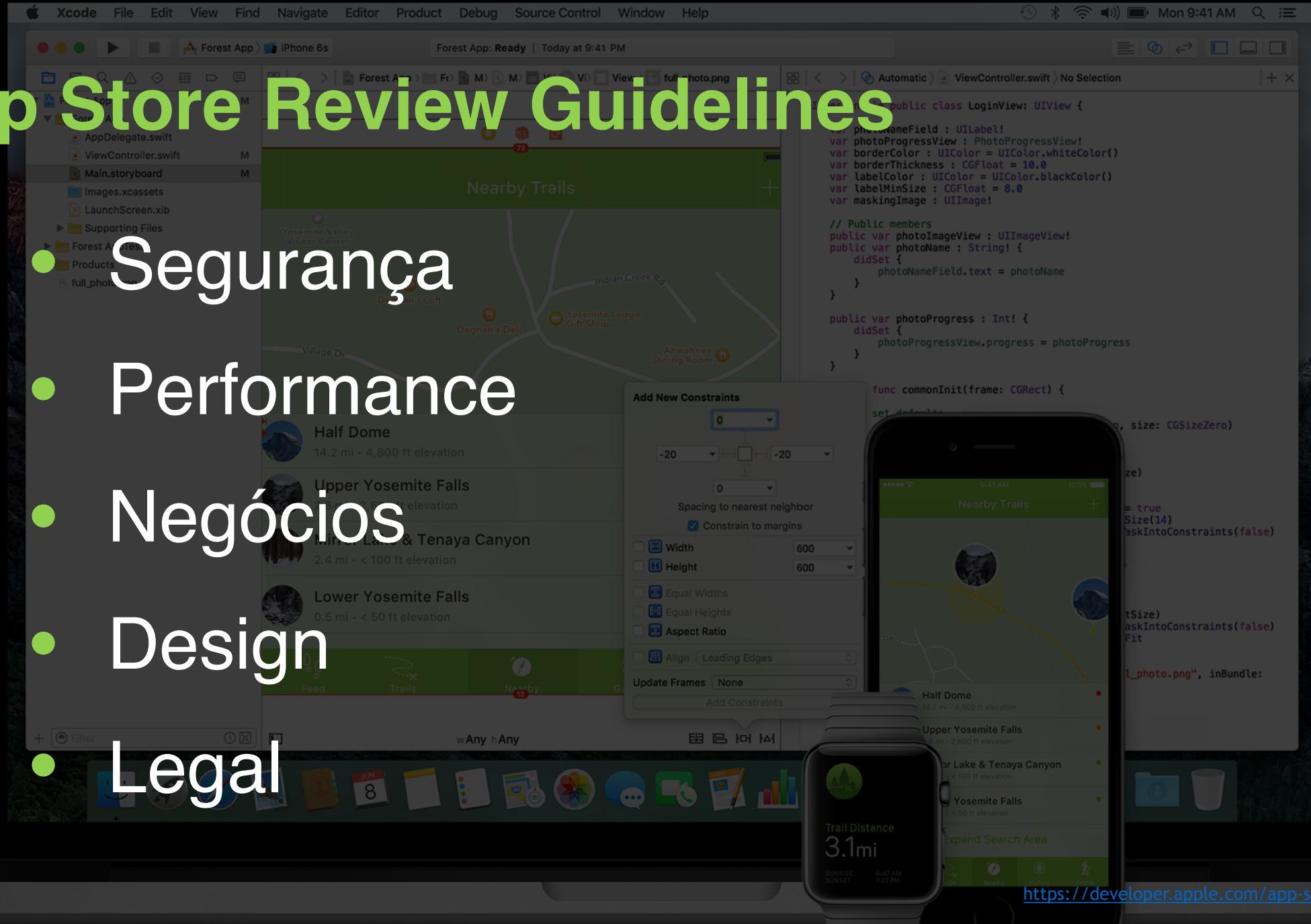
- Segurança

- Performance

- Negócios

- Design

- Legal



citi

<https://developer.apple.com/app-store/review/guidelines/>

13 de mai de 2016 às 18:54

De Apple

14. PERSONAL ATTACKS

14.3 Details

Your app enables users to post content anonymously but does not have the required precautions in place.

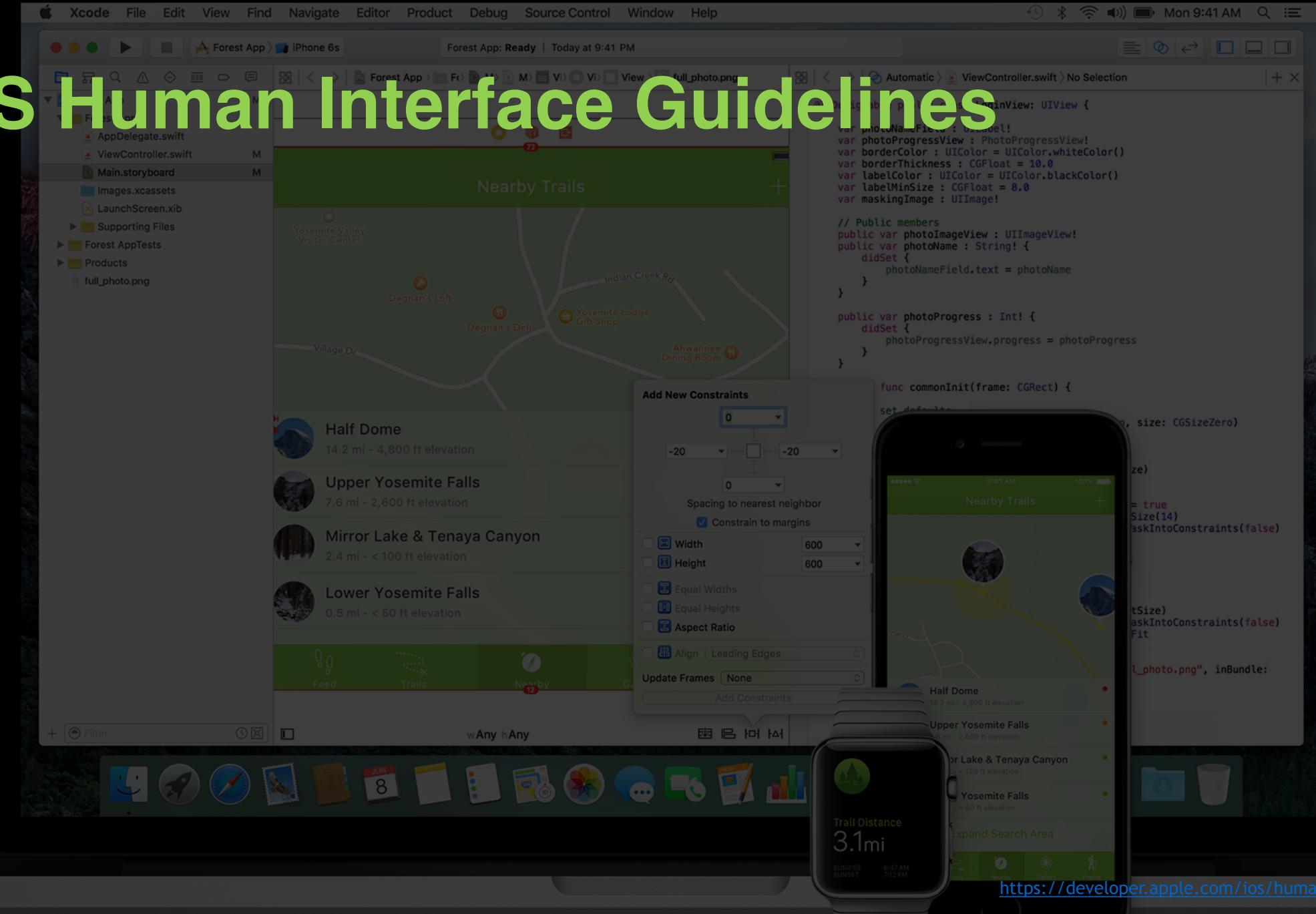
Next Steps

It is necessary that you put all of the following precautions in place:

- Age rating must reflect 17+
- The ability to report a post
- The ability to block a user
- The ability to immediately remove a post from the feed
- Contact information available in the app for users to report inappropriate activity
- A backend mechanism for identifying and blocking users who violate terms and conditions



iOS Human Interface Guidelines



citi

<https://developer.apple.com/ios/human-interface-guidelines/>

iOS Human Interface Guidelines

- Aesthetic Integrity

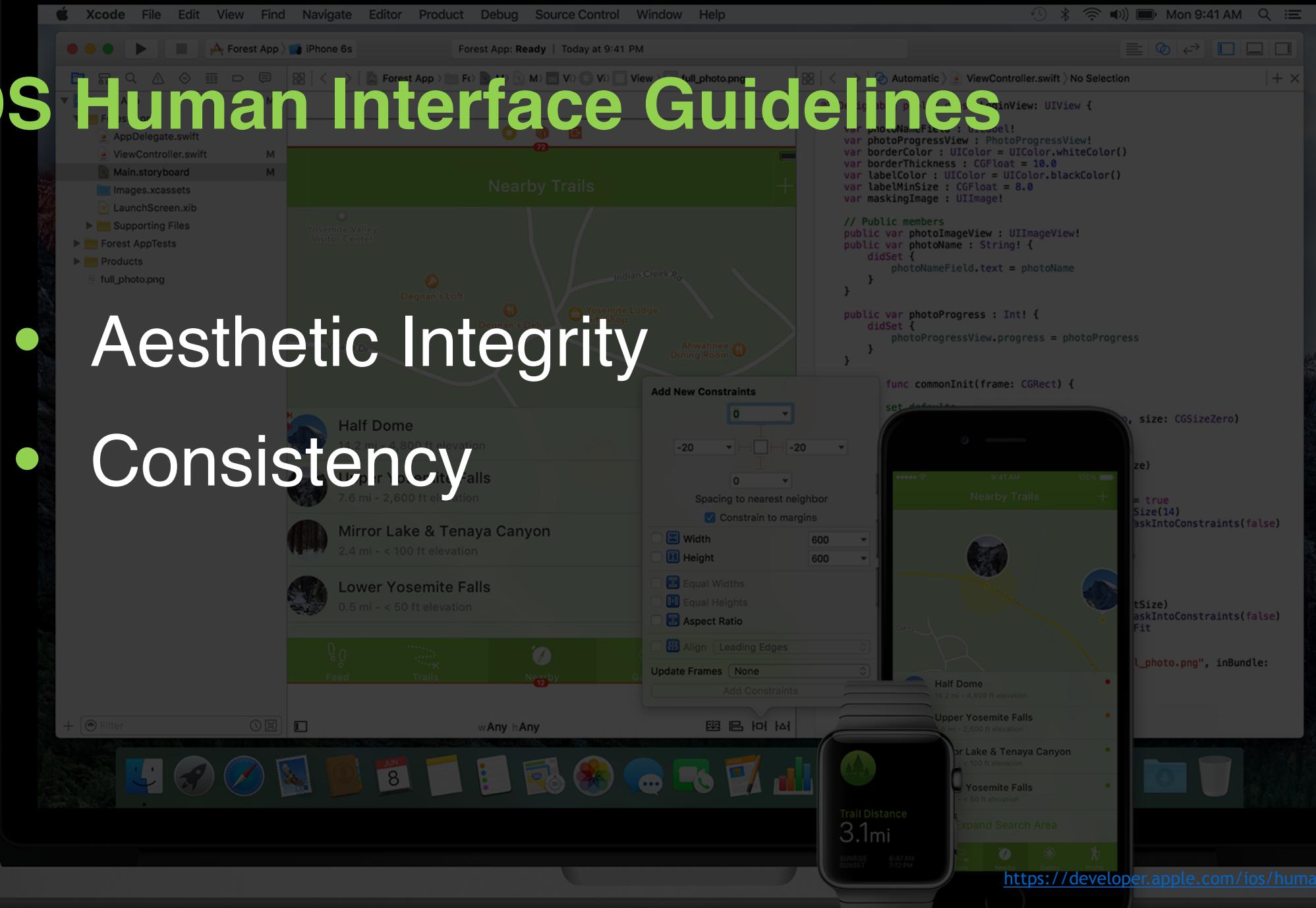


citi

<https://developer.apple.com/ios/human-interface-guidelines/>

iOS Human Interface Guidelines

- Aesthetic Integrity
- Consistency

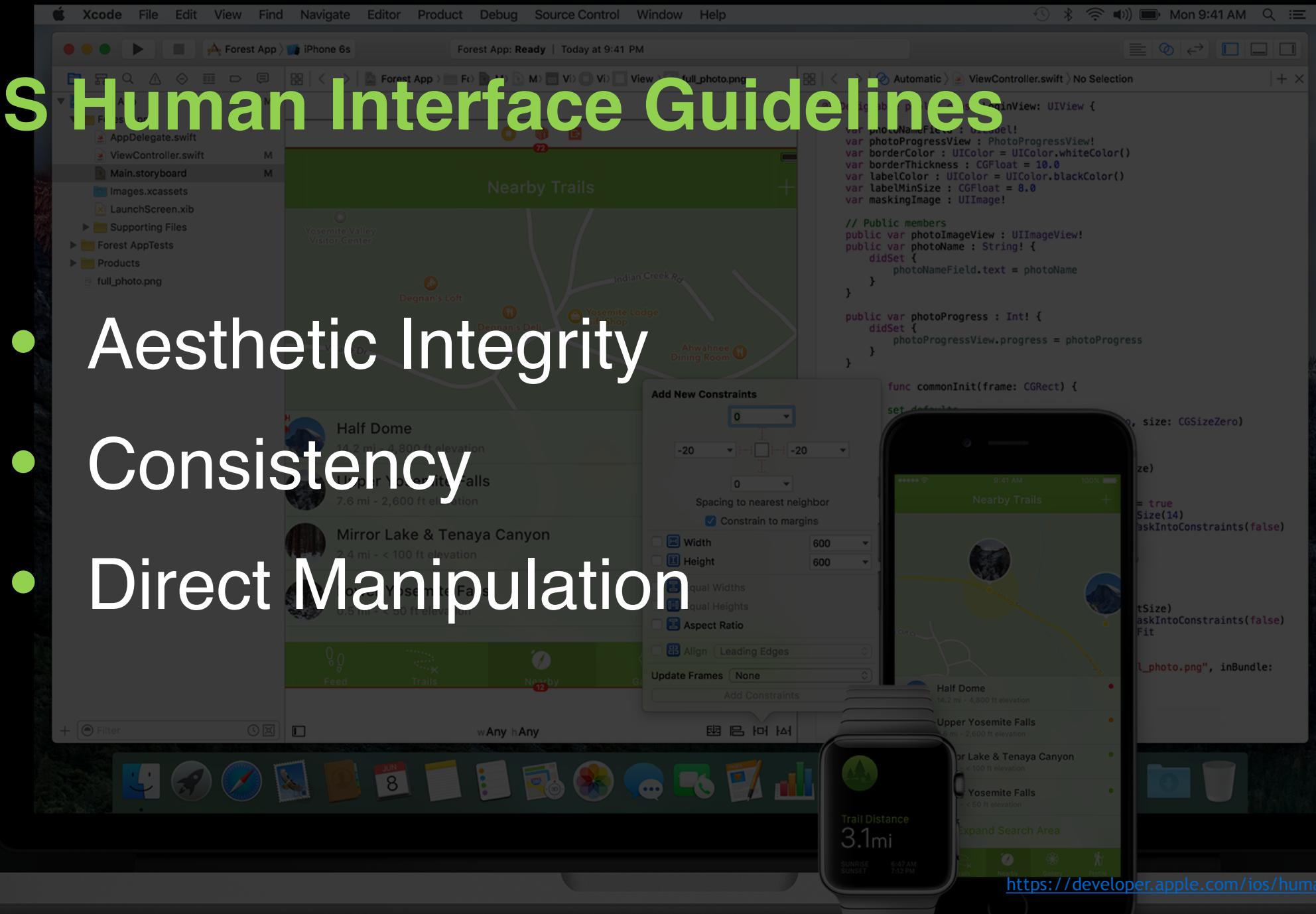


citi

<https://developer.apple.com/ios/human-interface-guidelines/>

iOS Human Interface Guidelines

- Aesthetic Integrity
- Consistency
- Direct Manipulation

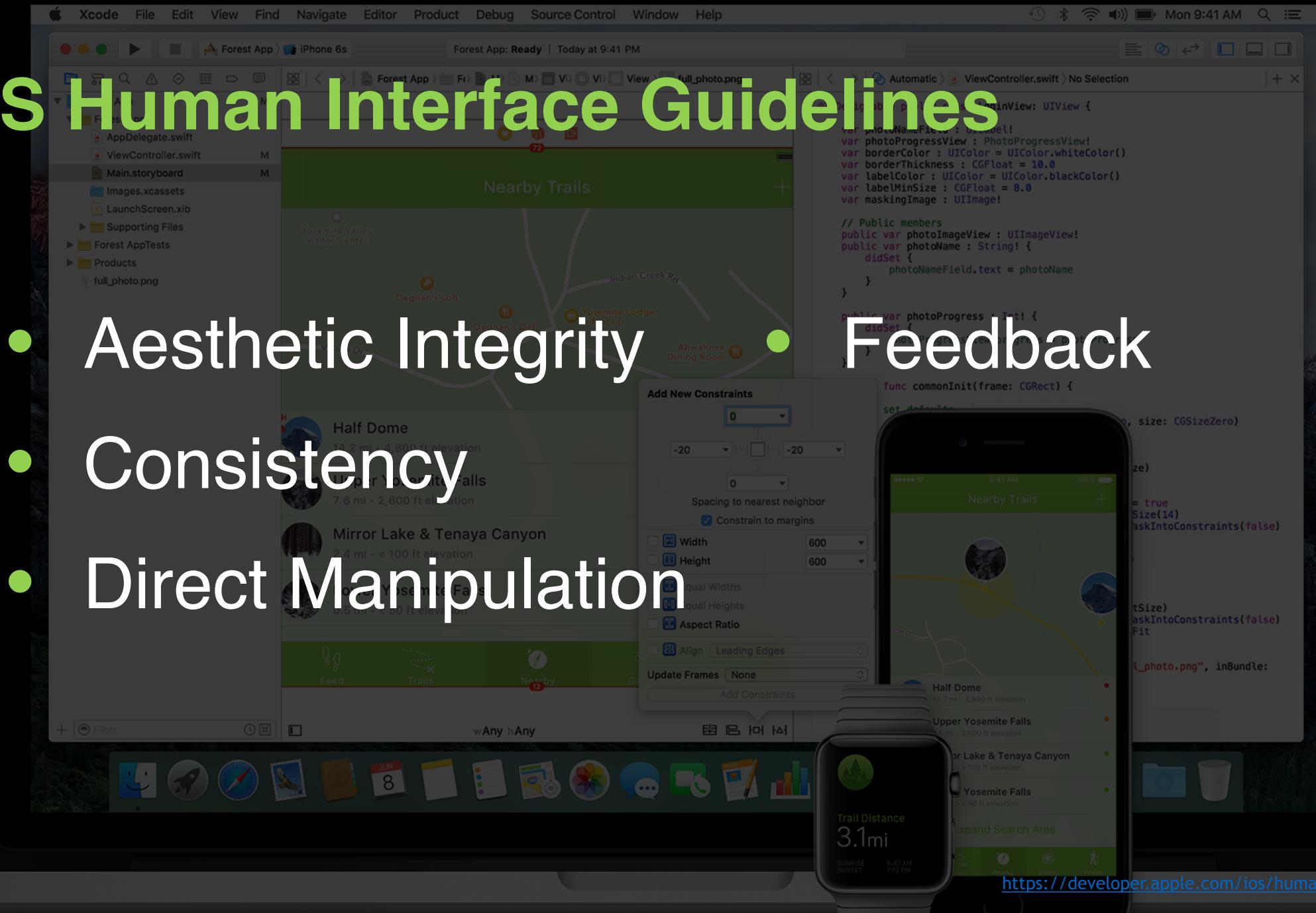


citi

<https://developer.apple.com/ios/human-interface-guidelines/>

iOS Human Interface Guidelines

- Aesthetic Integrity
- Consistency
- Direct Manipulation
- Feedback

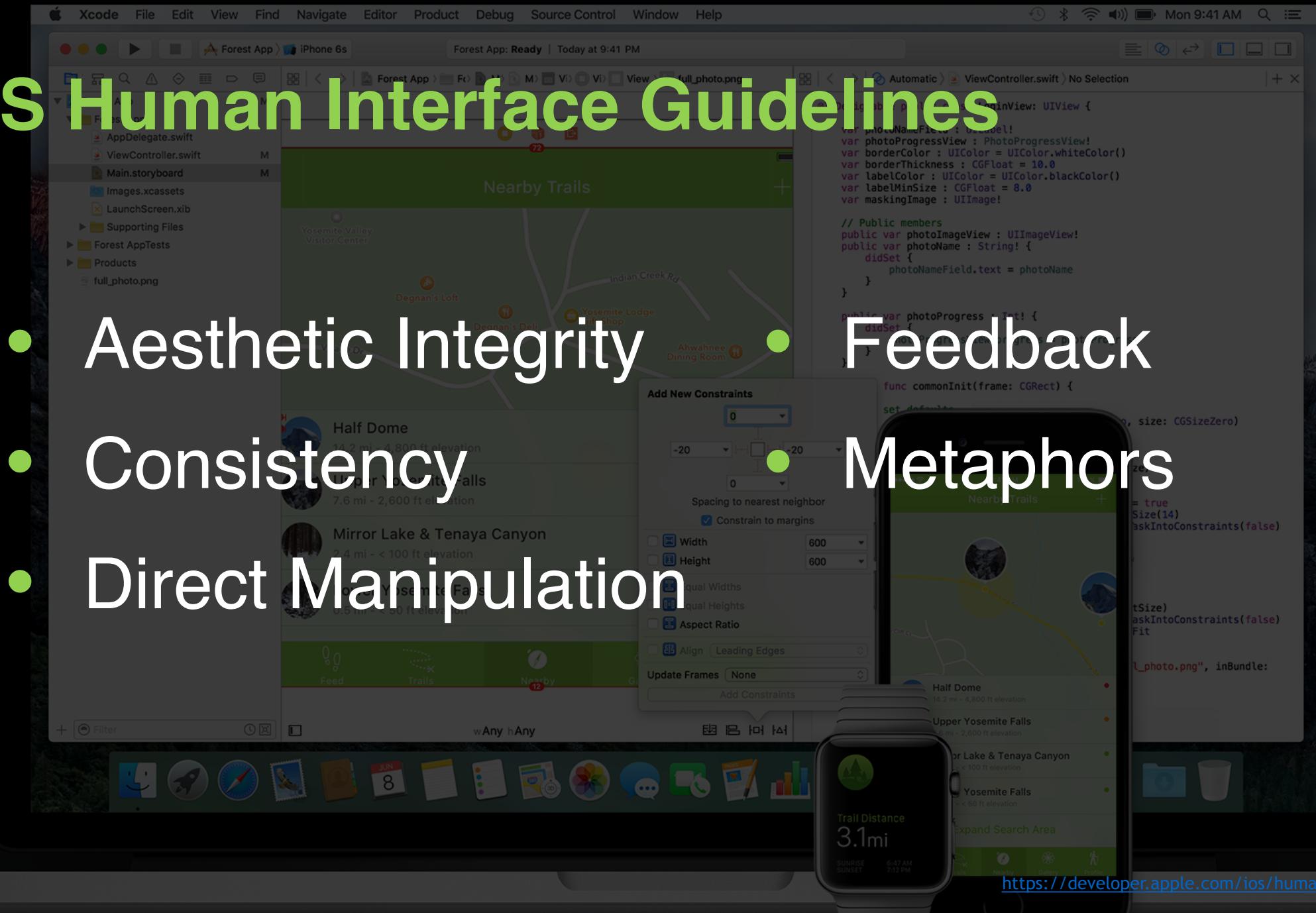


citi

<https://developer.apple.com/ios/human-interface-guidelines/>

iOS Human Interface Guidelines

- Aesthetic Integrity
- Consistency
- Direct Manipulation
- Feedback
- Metaphors

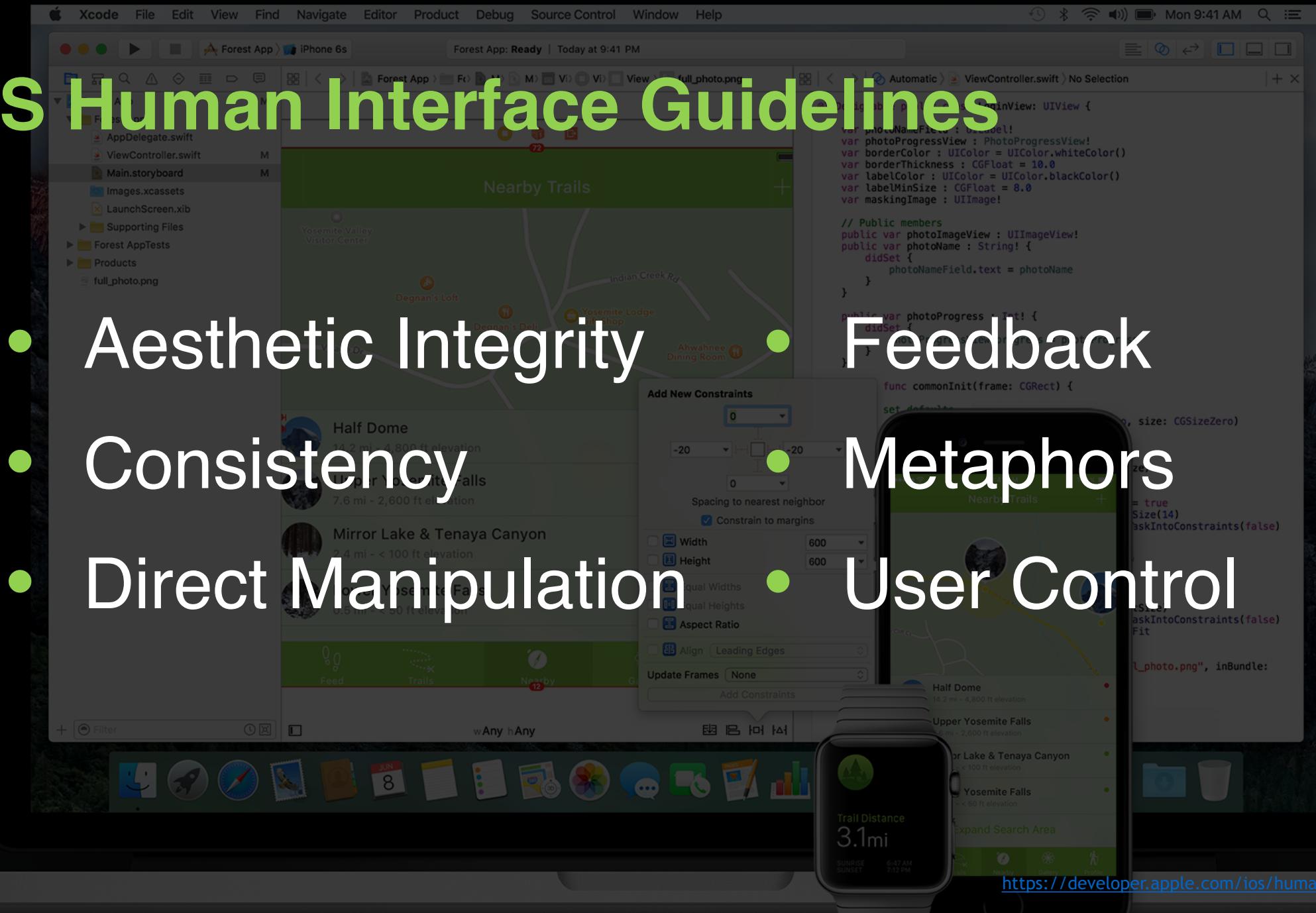


citi

<https://developer.apple.com/ios/human-interface-guidelines/>

iOS Human Interface Guidelines

- Aesthetic Integrity
- Consistency
- Direct Manipulation
- Feedback
- Metaphors
- User Control



citi

<https://developer.apple.com/ios/human-interface-guidelines/>

Objective

C



citi

Objective-C vs Swift



Objective-C vs Swift

- Tradicional
- Nova

Objective-C vs Swift

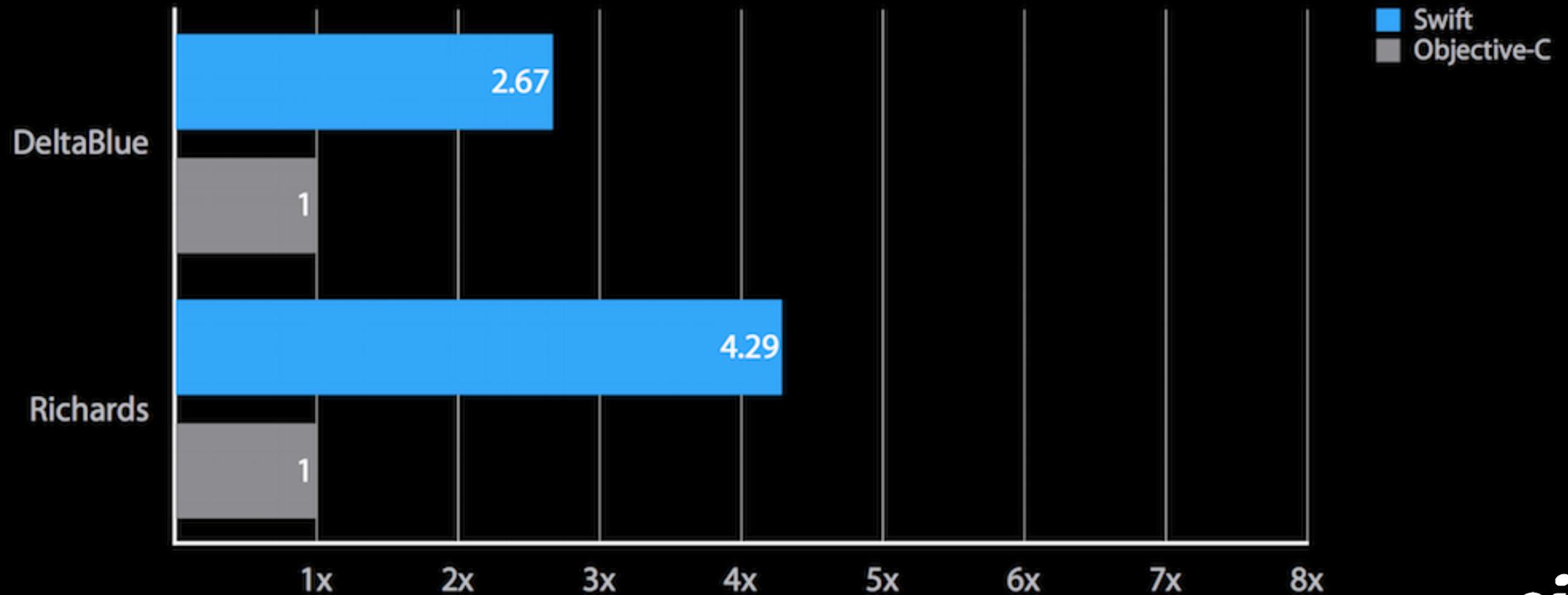
- Tradicional
- Menos Mudanças
- Nova
- Open source

Objective-C vs Swift

- Tradicional
- Menos Mudanças
- Mais Adotada
- Nova
- Open source
- Adotada pela Apple

Swift vs. Objective-C

Program speed (higher is better)







```
if ([delegate respondsToSelector:  
    @selector(application:willFinishLaunchingWithOptions:)]) {  
    [delegate application:app  
        willFinishLaunchingWithOptions:options];  
}
```



```
delegate.application?(app,  
    willFinishLaunchingWithOptions:options)
```



Swift



Xcode

Newton's Cradle and UIKit Dynamics

This playground uses UIKit Dynamics to create a [Newton's Cradle](#). Commonly seen on desks around the world, Newton's Cradle is a device that illustrates conservation of momentum and energy.

Let's create an instance of our UIKit Dynamics based Newton's Cradle by adding more colors to the array to increase the number of balls in the cradle.

```
let newtonsCradle = NewtonsCradle(colors: [red, blue, green, yellow])
```

Size and spacing

Try changing the size and spacing of the balls and see how that changes the behavior of the cradle. What happens if you make ballPadding a negative number?

```
newtonsCradle.ballSize = CGSize(width: 60, height: 60)  
newtonsCradle.ballPadding = 2.0
```

Behavior

Adjust elasticity and resistance to change how the balls react to each other.

```
newtonsCradle.itemBehavior.elasticity = 1.0  
newtonsCradle.itemBehavior.resistance = 0.2
```

Shape and rotation



Swift Playgrounds

// Comentários em Swift

```
/*
```

Comentários
com várias linhas

```
*/
```

// Memória

// Variáveis

```
var nome = "Hilton"  
var idade: Int = 21
```

// Variáveis

```
var nome = "Hilton"  
var idade: Int = 21
```

Palavra-reservada
(keyword)

// Variáveis

```
var nome = "Hilton"  
var idade: Int = 21
```

Palavra-reservada Identificador
(keyword)

// Variáveis

```
var nome = "Hilton"  
var idade = 21
```

Palavra-reservada Identificador Tipo
(keyword)

// Variáveis

```
var nome = "Hilton"  
var idade: Int = 21
```

Palavra-reservada
(keyword)

Identificador

Tipo operador
de
atribuição

// Variáveis

```
var nome = "Hilton"  
var idade: Int = 21
```

Palavra-reservada <i>(keyword)</i>	Identificador	Tipo	Operador de atribuição	Valor (String) Valor (Int)
---------------------------------------	---------------	------	------------------------------	-------------------------------

// Atribuição de novo valor

idade = 22

Palavra-reservada <i>(keyword)</i>	Identificador	Tipo	Operador de atribuição	Valor (Int)



// Atribuição de novo valor

idade = 22

Palavra reservada
(keyword)

Identificador

Operador
de
atribuição

Valor (Int)



// Constantes

```
let diaNascimento = 2  
let nacionalidade: String = "Brasileiro"
```

// Constantes

```
let diaNascimento = 2  
let nacionalidade: String = "Brasileiro"
```

Palavra-reservada
(keyword)

// Constantes

```
let diaNascimento = 2  
let nacionalidade: String = "Brasileiro"
```

Palavra-reservada
(keyword)

Identificador



// Constantes

```
let diaNascimento = 2  
let nacionalidade: String = "Brasileiro"
```

Palavra-reservada
(keyword)

Identificador

Tipo

// Constantes

```
let diaNascimento = 2  
let nacionalidade: String = "Brasileiro"
```

Palavra-reservada
(keyword)

Identificador

Tipo

Operador
de
atribuição



// Constantes

```
let diaNascimento = 2  
let nacionalidade: String = "Brasileiro"
```

Palavra-reservada
(keyword)

Identificador

Tipo

Operador
de
atribuição

Valor (Int)
Valor (String)



// Tipos

// Tipos numéricos

```
let inteiro: Int = 10
```

```
let decimal: Double = 3.14159265359
```

```
let pi: Float = 3.14159265359
```

Valor (Int)

Valor (Double)

Valor (Float)



// Tipos numéricos

```
let inteiro: Int = 10
let decimal: Double = 3.14159265359
let pi: Float = 3.14159265359
                                //arredondado para 3.141593
```

// Booleans

```
let souRecifense: Bool = true  
var odeioSwift = false
```

Valor (Bool)



// Strings

let sobrenome: String = "Pintor"

let cpf = "111-404-222.16"

Valor (String)



// Tuplas

```
let estado: (String, String) = ("Recife", "PE")
```

// Tuplas

```
let estado: (String, String) = ("Recife", "PE")  
estado.0 // acessando primeiro valor da tupla estado
```

Identificador . Índice
Ponto



// Tuplas

```
let rg = (numero: 9153865, orgao: "SDS", UF: "PE")
```

// Tuplas

```
let rg = (numero: 9153865, orgao: "SDS", UF: "PE")  
rg.numero // acessando valor "numero" da tupla rg
```

Identificador . Label
Ponto



// Optionals

```
var sentidoVida: Int?  
sentidoVida // nesse ponto não tem valor (nil)
```

Tipo ?
Interrogação



// Optionals

```
sentidoVida = 42
```

```
sentidoVida // nesse ponto o valor é 42
```

// Forced unwrapping

```
var titulo: String?  
titulo = "PhD"  
var unwrapped = titulo! // extrai o valor do opcional
```

titulo é do tipo **String?**
unwrapped é do tipo **String**



// Optional binding

```
var opcional: Bool?  
opcional = true
```

```
if let concreto = opcional {  
    concreto // valor extraído: true  
}
```

opcional é do tipo Bool?
concreto é do tipo Bool



// Operadores

// Atribuição

```
nome = "Hilton"  
idade = 21
```

// Aritmética

```
let dez: Int = 10  
let dois: Int = 2
```

// Aritmética

dez + dois	// 12
dez - dois	// 8
dez / dois	// 5
dez * dois	// 20

dez e dois são do mesmo tipo

// Resto

5 % 2 // 1

5 % 5 // 0

2 % 5 // 2

// Resto

5

2



// Resto

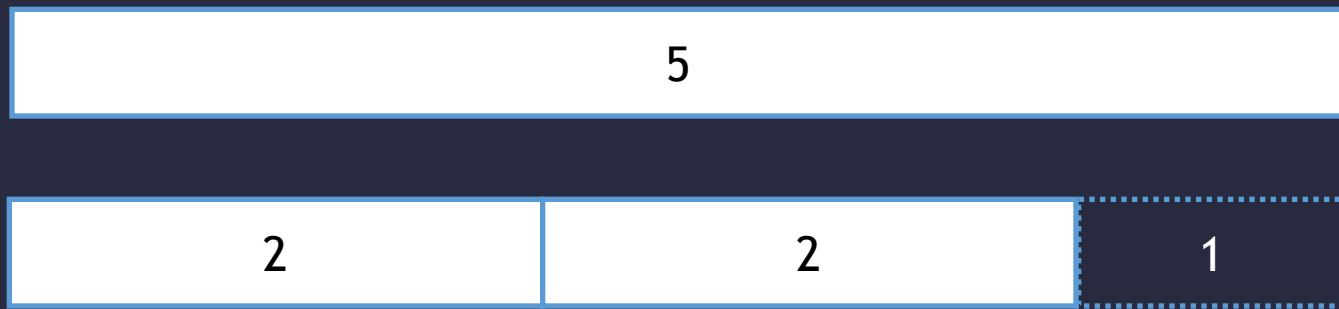
5

2

2



// Resto



$$5 \% 2 \quad // 1$$

// Concatenação

```
"olá" + "mundo" // resulta em: "olámundo"  
"oi " + nome // resulta em: "oi Hilton"
```

operador **+** realiza operações diferentes de acordo
com o **tipo dos operandos (Int, String, ...)**

// Comparação

```
2 === 2 // dois é igual a dois? true  
10 < 1 // 10 é menor que 1? false
```

// Comparação

nome != "Hilton"

/* o valor de nome é diferente de
"Hilton"? false */

idade >= 21

/* o valor de idade é maior ou igual a
21? true */

// Negação

odeioSwift = false

!odeioSwift // retorna true



// Conjunção (*And*)

```
true && true // retorna true  
false && true // retorna false  
false && false // retorna false
```

// Disjunção (*Or*)

```
true || true    // retorna true  
false || true   // retorna true  
false || false  // retorna false
```

// Intervalos

```
0...3 // representa 0, 1, 2, 3  
0..<3 // representa 0, 1, 2
```

// Coleções

// Arrays

```
var cidades = ["Recife", "Olinda"]  
cidades[0] // retorna "Recife"
```

// Arrays

```
var cidades = [0: "Recife", 1: "Olinda"]  
cidades[0] // retorna "Recife"
```

// Arrays - adição

```
cidades.append( "Jaboatão")
cidades[2] // retorna "Jaboatão"
```

// Arrays - adição

```
0 "Recife" 1 "Olinda" .append( "Jaboatão")
cidades[2] // retorna "Jaboatão"
```

// Arrays - adição

```
0 "Recife" 1 "Olinda" .append( 2 "Jaboatão" )  
cidades[2] // retorna "Jaboatão"
```

// Arrays - remoção

0	"Recife"	1	"Olinda"	2	"Jaboatão"
---	----------	---	----------	---	------------

`cidades.removeLast()` // remove "Jaboatão"

`cidades.removeFirst()` // remove "Recife"

`cidades.removeAll()` // remove todos os elementos

// Arrays - remoção

0	"Recife"	1	"Olinda"
---	----------	---	----------

`cidades.removeLast()` // remove "Jaboatão"

`cidades.removeFirst()` // remove "Recife"

`cidades.removeAll()` // remove todos os elementos

// Arrays - remoção



`cidades.removeLast()` // remove "Jaboatão"

`cidades.removeFirst()` // remove "Recife"

`cidades.removeAll()` // remove todos os elementos

// Arrays - remoção

`cidades.removeLast()` // remove "Jaboatão"

`cidades.removeFirst()` // remove "Recife"

`cidades.removeAll()` // remove todos os elementos

// Arrays - contagem

```
var cidadesVisitadas: [String] = []
cidadesVisitadas.count // retorna 0 (zero)
```



// Arrays - contagem

```
cidadesVisitadas.append("Recife")  
cidadesVisitadas.count // retorna 1
```

0	"Recife"
---	----------

// Arrays - modificação

```
cidadesVisitadas[0] = "Olinda"  
cidadesVisitadas // ["Olinda"]
```

0	"Recife"
---	----------

// Arrays - modificação

```
cidadesVisitadas[0] = "Olinda"  
cidadesVisitadas // ["Olinda"]
```

0	"Olinda"
---	----------

// Arrays - acesso inválido

```
cidadesVisitadas[cidadesVisitadas.count]
cidades[-1]
```

0	"Olinda"
---	----------

// Arrays - acesso inválido

cidadesVisitadas[cidadesVisitadas.count]
cidades[-1]

0	"Olinda"
---	----------

// Dicionários

```
var capitais: [String: String] = ["PE": "Recife"]
capitais["PE"] // retorna "Recife"
```

Chave	Valor
“PE”	“Recife”
...	nil

// Dicionários - adição

```
capitais["PB"] = "João Pessoa"
```

```
capitais // ["PB": "João Pessoa", "PE": "Recife"]
```

Chave	Valor
"PE"	"Recife"
"PB"	"João Pessoa"
...	nil

// Dicionários - modificação

```
capitais["PB"] = "Maria Pessoa"  
capitais // ["PB": "Maria Pessoa", "PE": "Recife"]
```

Chave	Valor
“PE”	“Recife”
“PB”	“João Pessoa”
...	nil

// Dicionários - modificação

```
capitais["PB"] = "Maria Pessoa"  
capitais // ["PB": "Maria Pessoa", "PE": "Recife"]
```

Chave	Valor
"PE"	"Recife"
"PB"	"Maria Pessoa"
...	nil

// Dicionários - remoção

```
capitais["PB"] = nil  
capitais.removeValue(forKey: "PE")
```

Chave	Valor
“PE”	“Recife”
“PB”	“João Pessoa”
...	nil

// Dicionários - remoção

```
capitais["PB"] = nil  
capitais.removeValue(forKey: "PE")
```

Chave	Valor
“PE”	“Recife”
“PB”	nil
...	nil

// Dicionários - remoção

```
capitais["PB"] = nil  
capitais.removeValue(forKey: "PE")
```

Chave	Valor
“PE”	nil
“PB”	nil
...	nil

// Dicionários - contagem

```
var dictVazio: [Int: String] = [:]  
dictVazio.count // retorna 0
```

Chave	Valor
...	nil

// Dicionários - acesso sem valor

```
dictVazio[0] // retorna nil  
dictVazio[-1] // retorna nil
```

Chave	Valor
0	nil
-1	nil
...	nil

// Controle de Fluxo



// Desvios Condicionais

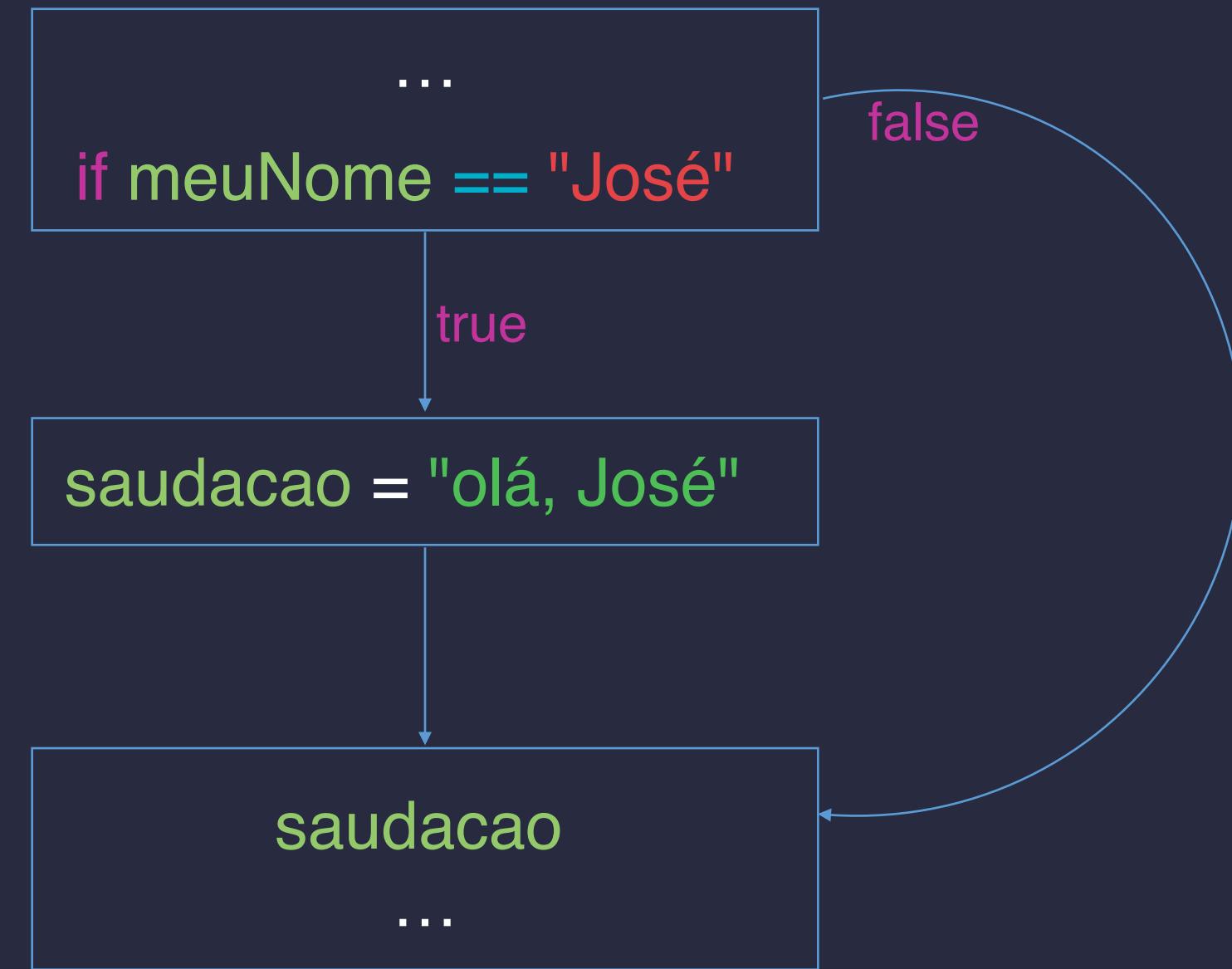
// If (se)

```
var meuNome = "José"
var saudacao = ""

if meuNome == "José" {
    // executar caso verdadeiro:
    saudacao = "Olá, José"
}

saudacao // tem valor de "Olá, José"
```

// If (se)

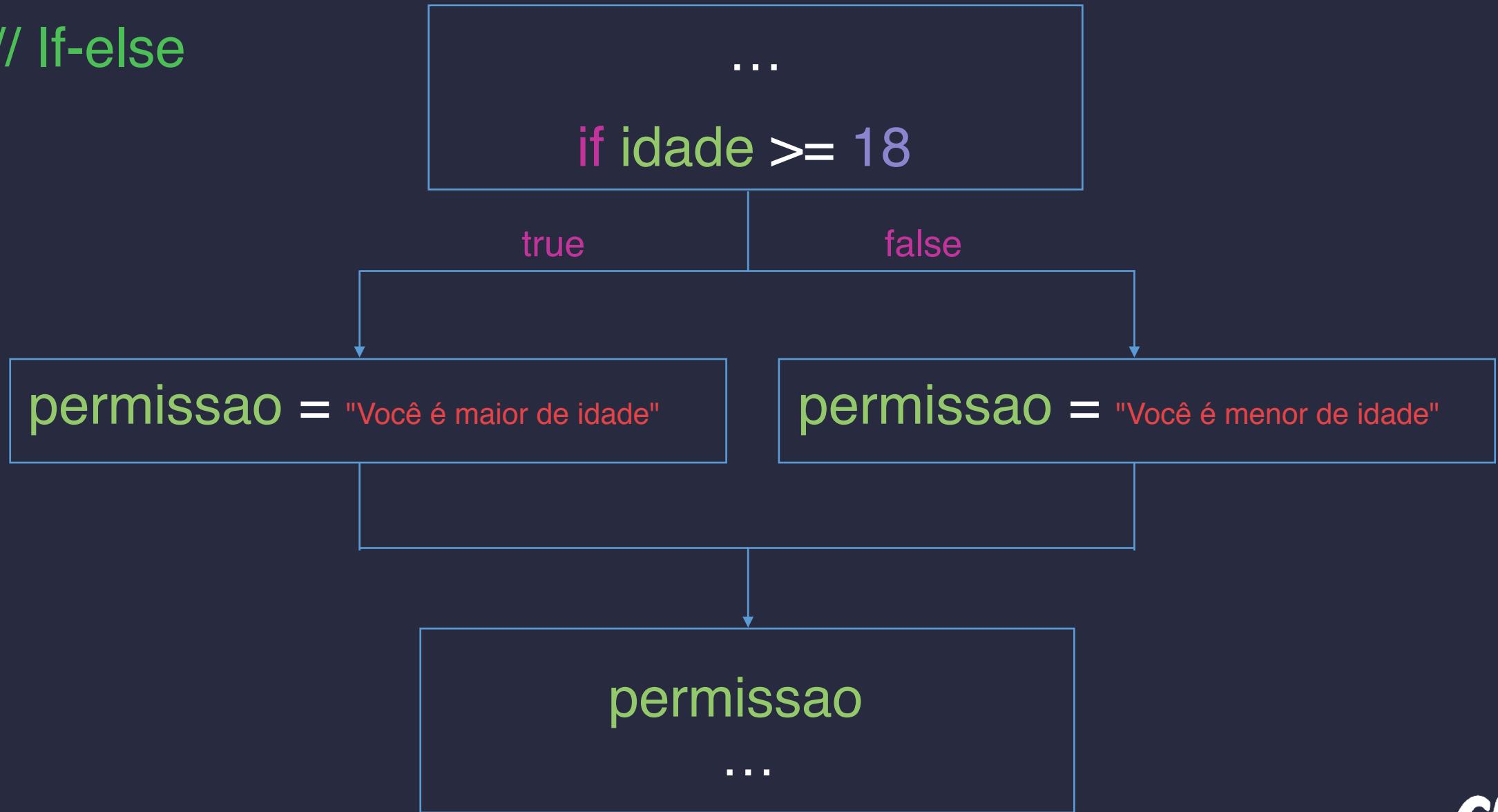


```
// If-else    idade = 17
var permissao = ""

if idade >= 18 {
    permissao = "Você é maior de idade"
} else {
    // executar caso falso:
    permissao = "Você é menor de idade"
}

permissao // tem valor de "Você é menor de
idade"
```

// If-else



```
// else if
idade = 16
var mensagem = ""

if idade >= 18 {
    mensagem = "você é um adulto"
} else if idade >= 12{
    mensagem = "você é um adolescente"
} else {
    mensagem = "você é uma criança"
}

mensagem // tem valor de "você é um adolescente"
```

// If-else

...
if idade >= 18

true

false

mensagem = "você é um adulto"

else if idade >= 12

true

false

mensagem = "você é um
adolescente"

mensagem = "você é uma
criança"

permissao

...

// switch

```
var animal = "Cachorro"
var som: String

switch animal {

    case "Gato":
        som = "miado"

    case "Cachorro":
        som = "latido"

    case "Leão":
        som = "rugido"

    default:
        // se for qualquer outro animal
        som = "indefinido"
}

som // tem valor de "latido"
```

// switch

```
animal = "Gata"
som = ""

switch animal {

    case "Gato", "Gata":
        som = "miado"

    case "Cachorro", "Cadela":
        som = "latido"

    case "Leão", "Leoa":
        som = "rugido"

    default:
        // se for qualquer outro animal
        som = "indefinido"
}

som // tem valor de "miado"
```

// Loops

// for in

```
cidadesVisitadas = ["Recife", "Olinda", "Jaboatão"]
mensagem = "nós visitamos:"

for cidade in cidadesVisitadas {
    mensagem = mensagem + " " + cidade
}

mensagem // tem valor de "nós visitamos: Recife Olinda Jaboatão"
```

// for in

```
for cidade in cidadesVisitadas {  
    mensagem = mensagem + " " + cidade  
}
```

Execução	Valor de cidade	Valor de mensagem
Antes do loop	Não existe	nós visitamos:
Fim da 1a Iteração	Recife	nós visitamos: Recife
Fim da 2a Iteração	Olinda	nós visitamos: Recife Olinda
Fim da 3a Iteração	Jaboatão	nós visitamos: Recife Olinda Jaboatão
Após o loop	Não existe	nós visitamos: Recife Olinda Jaboatão

cidadesVisitadas = ["Recife", "Olinda", "Jaboatão"]



// for in range

```
var quadrados: [Int] = []  
  
for numero in 1...10 {  
    let quadrado = numero * numero  
    quadrados.append(quadrado)  
}  
  
quadrados // tem valor de [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

// while (*enquanto*)

```
var preco = 100
var devoComprar = false

//enquanto o preço for caro não devo comprar
while !devoComprar {

    preco = preco - 1 // preço diminui

    //se o preço chegar em 50, eu devo comprar.
    if preco == 50 {
        devoComprar = true
    }
}

preco          // tem valor de 50
devoComprar   // tem valor de true
```

// Funções e Closures

// funções - declaração

```
func maior(primeiro: Int, segundo: Int) -> Int {  
    if primeiro > segundo {  
        return primeiro  
  
    } else {  
        return segundo  
    }  
}
```

// funções - declaração

```
func maior(primeiro: Int, segundo: Int) -> Int {  
    if primeiro > segundo {  
        return primeiro  
    }
```

Palavra-reservada } else {
(keyword)
 return segundo
}
}

// funções - declaração

```
func maior(primeiro: Int, segundo: Int) -> Int {  
    if primeiro > segundo {  
        return primeiro  
    }  
    else {  
        return segundo  
    }  
}
```

Palavra-reservada } Identificador
(keyword)

// funções - declaração

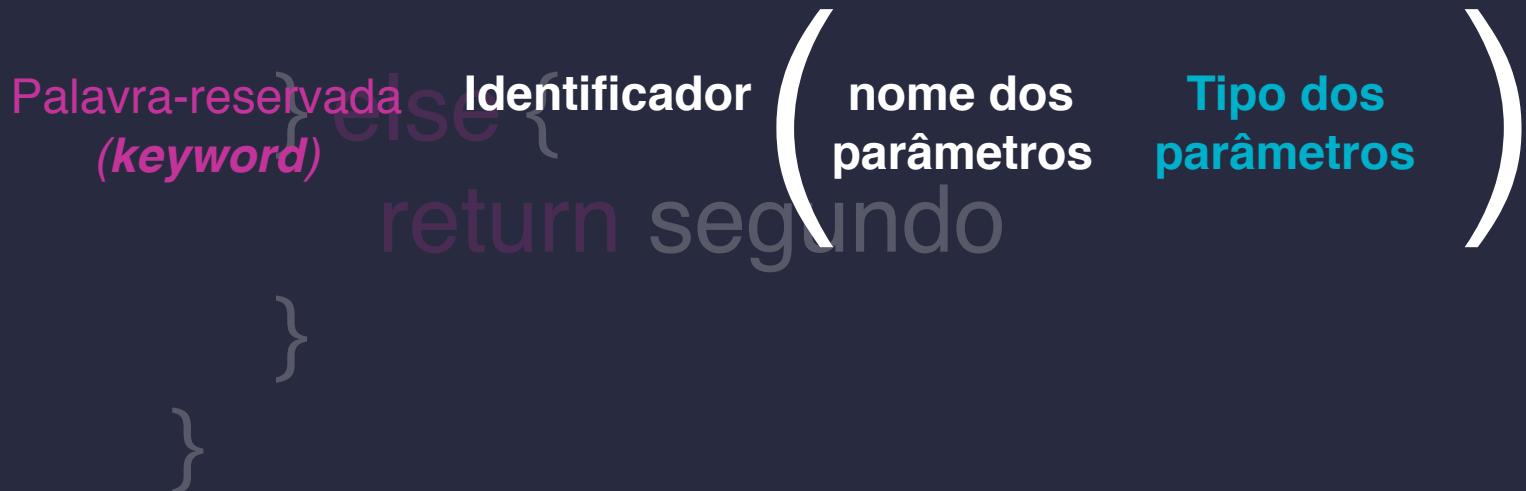
```
func maior(primeiro: Int, segundo: Int) -> Int {  
    if primeiro > segundo {  
        return primeiro  
    }  
    else {  
        return segundo  
    }  
}
```

Palavra-reservada } **else** { Identificador (keyword)

nome dos parâmetros

// funções - declaração

```
func maior(primeiro:Int, segundo:Int) -> Int {  
    if primeiro > segundo {  
        return primeiro  
    }  
    else {  
        return segundo  
    }  
}
```



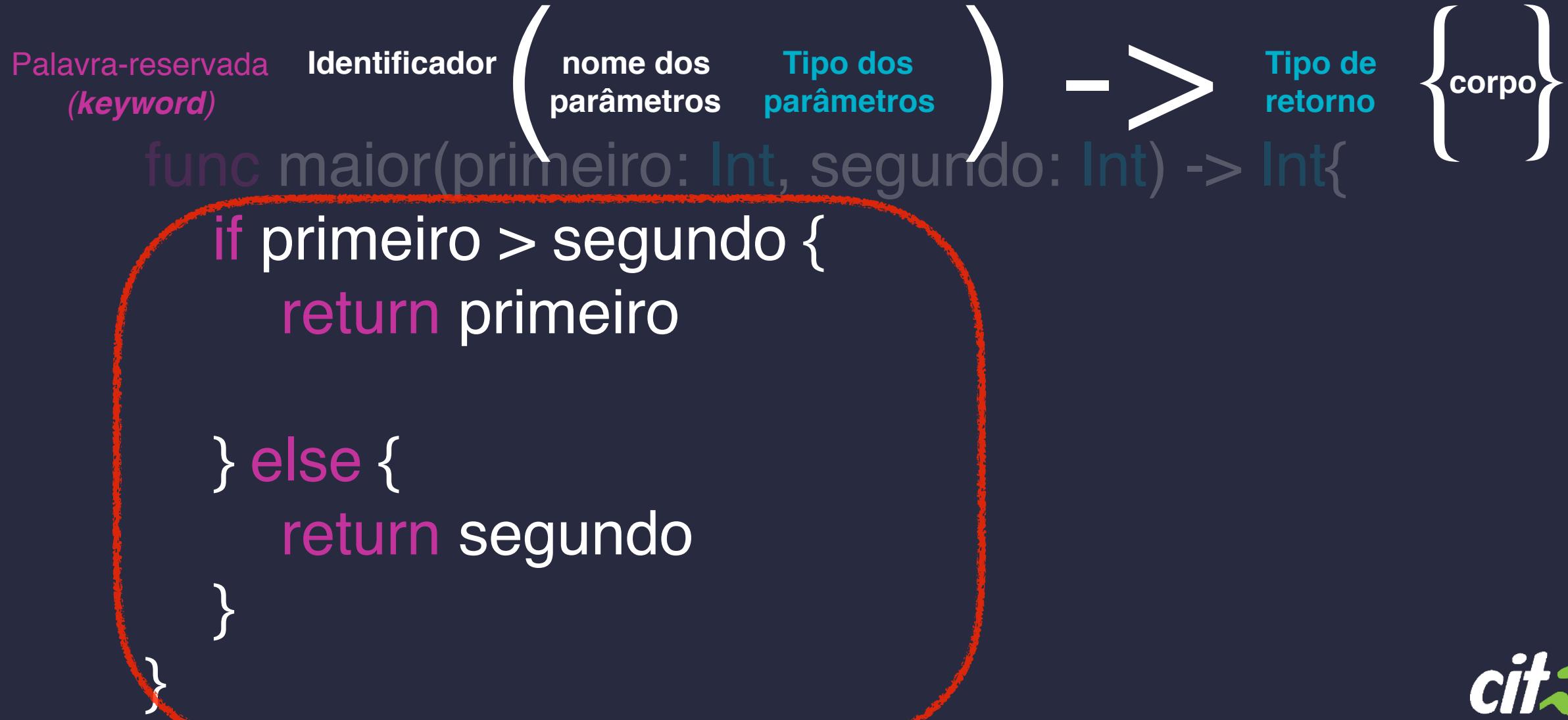
// funções - declaração

```
func maior(primeiro: Int, segundo: Int) -> Int {
```

```
    if primeiro > segundo {  
        return primeiro
```



// funções - declaração



// funções - chamada

```
let um = 1
```

```
let dois = 2
```

```
let maximo = maior(primeiro: um, segundo: dois)
```

```
maximo // tem valor de 2
```

// funções - chamada

```
let um = 1
```

```
let dois = 2
```

```
let maximo = maior(primeiro: um, segundo: dois)
```

```
maximo // tem valor de 2
```

// funções - chamada

```
let um = 1
```

```
let dois = 2
```

```
let maximo = maior(primeiro: um, segundo: dois)
```

maximo // tem valor de  2

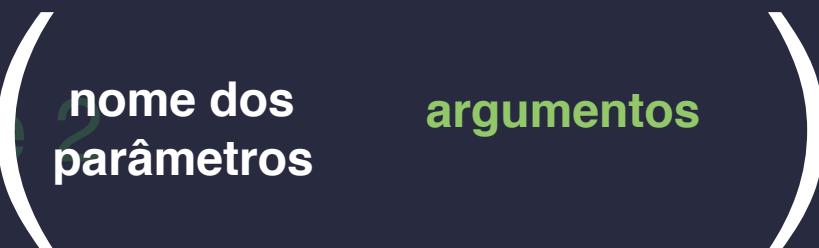
Identificador

nome dos
parâmetros

// funções - chamada

```
let um = 1  
let dois = 2
```

```
let maximo = maior(primeiro: um, segundo: dois)
```

maximo // tem valor de  (nome dos parâmetros argumentos)

Identificador

// funções são tipos

O **tipo** de uma função é dado pelo tipo de seus **parâmetros** e de seu **retorno**, separados por uma seta.

// funções são tipos

```
func maior(primeiro: Int, segundo: Int) -> Int{...}
```

// funções são tipos

func maior(primeiro: Int, segundo: Int) -> Int{...}

(primeiro: Int, segundo: Int) -> Int

// funções são tipos

func maior(primeiro: Int, segundo: Int) -> Int{...}

(primeiro: Int, segundo: Int) -> Int

(Int, Int) -> Int

// funções de primeira ordem

Funções que recebem como
parâmetro, ou **retornam**, outras
funções.

// funções de primeira ordem

```
var array = [1, 2, 3]
```

```
func somaUm(valor: Int) -> Int {  
    return valor + 1  
}
```

// map aplica a função somaUm a cada elemento de array

```
var novoArray = array.map(somaUm)
```

```
novoArray // tem valor de [2, 3, 4]
```

// funções de primeira ordem

```
func somaUm(valor: Int) -> Int {  
    return valor + 1  
}
```

(Int) -> Int

// map

```
var array = [1, 2, 3]
```

```
var novoArray = array.map(somaUm)
```



// map

```
var array = [1, 2, 3]
```

```
var novoArray = [ somaUm(valor: array[0]),      somaUm(valor: array[1]),      somaUm(valor: array[2]) ]
```

// closures

Funções **anônimas** que podem ser
definidas no mesmo lugar em que
são chamados

```
// map com closure
```

```
array = [1, 2, 3]
```

```
novoArray = []
```

```
novoArray = array.map({ (numero: Int) -> Int in  
    return numero + 1  
})
```

```
novoArray // tem valor de [2, 3, 4]
```

// closures

```
novoArray = array.map({(numero: Int) -> Int in  
    return numero + 1}
```

}

{ }
{ }

// closures

```
novoArray = array.map({ numero: Int) -> Int in  
    return numero + 1  
})
```

{ (nome dos
parâmetros) }

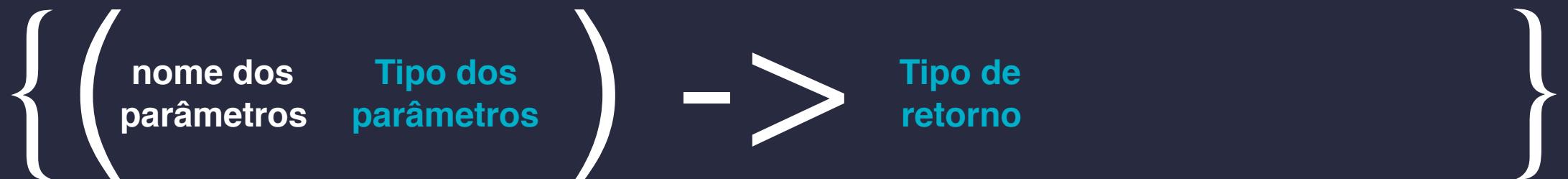
// closures

```
novoArray = array.map({ (numero:Int) -> Int in  
    return numero + 1  
})
```

{ (nome dos parâmetros Tipo dos parâmetros) }

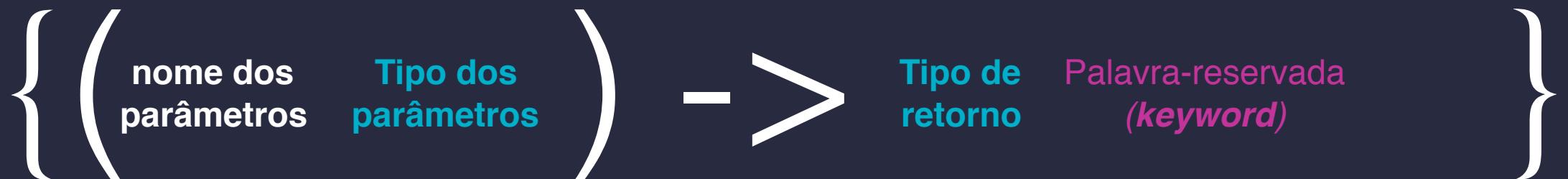
// closures

```
novoArray = array.map({ (numero: Int) -> Int in  
    return numero + 1  
})
```



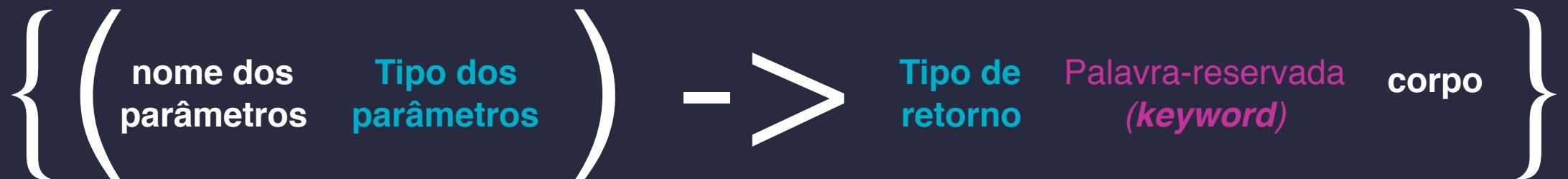
// closures

```
novoArray = array.map({ (numero: Int) -> Int  
    return numero + 1  
})
```



// closures

```
novoArray = array.map({ (numero: Int) -> Int in  
    return numero + 1  
})
```



// função vs closure

```
func somaUm(valor: Int) -> Int {  
    return valor + 1  
}
```

==

```
{ (numero: Int) -> Int in  
    return numero + 1  
})
```

// closure enxuto

```
novoArray = array.map({ (numero: Int) -> Int in  
    return numero + 1  
})
```

```
// closure enxuto
```

```
novoArray = array.map({ (numero: Int) -> Int in  
    return numero + 1  
})
```

// closure enxuto

```
novoArray = array.map({ numero in  
    return numero + 1  
})
```

```
// closure enxuto
```

```
novoArray = array.map({ numero in  
    return numero + 1  
})
```

```
// closure enxuto
```

```
novoArray = array.map({  
    return numero + 1  
})
```

```
// closure enxuto
```

```
novoArray = array.map({  
    return      $0 + 1  
})
```



// closure enxuto

```
novoArray = array.map({  
    $0 + 1  
})
```



// closure enxuto

```
novoArray = array.map({ $0 + 1 })
```



// closure enxuto

```
novoArray = array.map { $0 + 1 }
```



// closure enxuto

```
array = [1, 2, 3]
```

```
novoArray = []
```

```
novoArray = array.map {$0 + 1}
```

```
novoArray // tem valor de [2, 3, 4]
```

// função vs closure II

```
func somaUm(valor: Int) -> Int {  
    return valor + 1  
}
```

==

```
{ (numero: Int) -> Int in  
    return numero + 1  
})
```

==

```
{$0 + 1}
```

// Exercícios

// Exercício 01-a

Ano Bissexto

Chama-se ano bissexto o ano ao qual é acrescentado um dia extra, ficando ele com 366 dias.

Ocorrendo a cada **quatro anos** (exceto anos **múltiplos de 100** que não são **múltiplos de 400**).

Dado um ano como entrada,
identifique se o ano é
bissexto.

// Exercício 01-b

Ano Bissexto

Chama-se ano bissexto o ano ao qual é acrescentado um dia extra, ficando ele com 366 dias.

Ocorrendo a cada **quatro anos** (exceto anos **múltiplos de 100** que não são **múltiplos de 400**).

Calcule os anos bissextos de 1988 a 2018

// Exercício 01-c

Ano Bissexto

Chama-se ano bissexto o ano ao qual é acrescentado um dia extra, ficando ele com 366 dias.

Ocorrendo a cada **quatro anos** (exceto anos **múltiplos de 100** que não são **múltiplos de 400**).

Filtre um array de anos,
deixando somente os
bissextos