



PROF. RENÊ XAVIER

---

# DESENVOLVIMENTO PARA IOS 11 COM SWIFT 4

ONDE ENCONTRAR O MATERIAL?

[HTTPS://GITHUB.COM/RENEFX/IOS-2018-01](https://github.com/renefx/ios-2018-01)

---

**GITHUB**  
ALÉM DO TRADICIONAL BLACKBOARD DO IESB

O QUE VAMOS FAZER HOJE?

---

## AGENDA

- ▶ Banco de dados Realm





---

# USANDO O REALM

ANTES DE TUDO,  
VEJA A DOCUMENTAÇÃO

[HTTPS://REALM.IO/DOCS/SWIFT/LATEST/](https://realm.io/docs/swift/latest/)

# USANDO O REALM

---

## ADICIONAR O REALM NO PROJETO

- ▶ Se você instalou seu CocoaPods tem um certo tempo, talvez seja bom executar:

**pod repo update**

- ▶ Se não tivermos configurado o CocoaPods no nosso projeto:

**pod init**

- ▶ Após esse passo, podemos abrir nosso arquivo **Podfile** e:

- ▶ Verificar se a linha do **use\_frameworks** está descomentada

- ▶ Adicionar no arquivo o pod: **pod 'RealmSwift'**

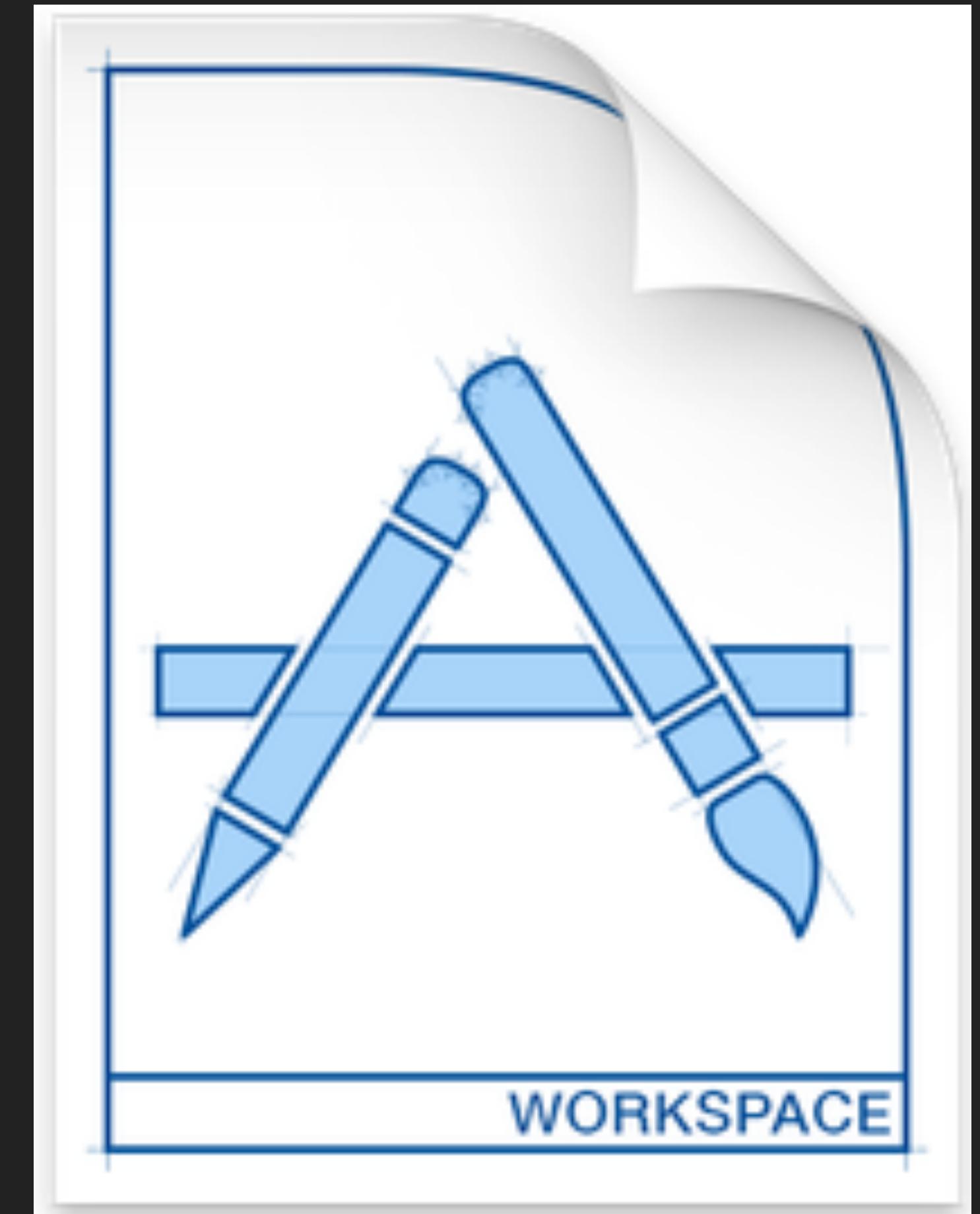
- ▶ Após salvar o arquivo, executamos o:

**pod install**

**ADICIONADO!**  
**SIMPLES!**

ABRA O XCWORKSPACE

VAMOS PASSAR A SÓ ABRIR O XCWORKSPACE



ELE TRABALHA  
COM OBJETOS,  
MAS COMO?

## COMO O REALM TRABALHA

- ▶ Alteramos nossa Model:
  - ▶ Importando a biblioteca RealmSwift
  - ▶ Indicando que ela e as properties dela devem ser mapeados pelo Realm
- ▶ Pegamos uma instância do Realm
- ▶ Utilizamos essa instância com o método que queremos passando o objeto que queremos
- ▶ Se alguma outra instância mudar o valor do seu objeto, ele já é atualizado. Sim parece mágica

VAMOS  
ALTERAR A  
NOSSA MODEL?

# USANDO O REALM

```
→ import RealmSwift ↓  
class Usuario: Object {  
    → @objc dynamic var nome = ""  
    @objc dynamic var idade = 0  
}
```

# USANDO O REALM

---

## TIPOS ACEITOS NA MODEL

- ▶ Int - RealmOptional
- ▶ Int8 - RealmOptional
- ▶ Int16 - RealmOptional
- ▶ Int32 - RealmOptional
- ▶ Int64 - RealmOptional
- ▶ List<Object>
- ▶ Object
- ▶ Bool - RealmOptional
- ▶ Double - RealmOptional
- ▶ Float - RealmOptional
- ▶ String
- ▶ NSDate
- ▶ NSData

# USANDO O REALM

## O REALM TEM UMA ESPECIFICIDADE AO DECLARAR PROPERTIES NA NOSSA MODEL

- ▶ Para os casos que se deseja iniciar a property com um valor inicial:
- ▶ (Também é possível declarar sem especificar o tipo)

```
@objc dynamic var id: Int = 0
@objc dynamic var numero: Double = 0.0
@objc dynamic var nome: String = ""
@objc dynamic var data: Data = "a".data(using: String.Encoding.utf8)!
@objc dynamic var dia: Date = Date(timeIntervalSince1970: 1)
```

## USANDO O REALM

### O REALM TEM UMA ESPECIFICIDADE AO DECLARAR PROPERTIES NA NOSSA MODEL

- Quando a property pode ser nula:

```
@objc dynamic var nome: String? = nil  
@objc dynamic var objeto: ObjetoRealm? = nil  
@objc dynamic var data: Data? = nil  
@objc dynamic var dia: Date? = nil
```

# USANDO O REALM

## O REALM TEM UMA ESPECIFICIDADE AO DECLARAR PROPERTIES NA NOSSA MODEL

- ▶ Quando a property pode ser nula:

```
@objc dynamic var nome: String?  
@objc dynamic var objeto: ObjetoRealm?  
@objc dynamic var data: Data?  
@objc dynamic var dia: Date?
```

# USANDO O REALM

## O REALM TEM UMA ESPECIFICIDADE AO DECLARAR PROPERTIES NA NOSSA MODEL

- ▶ Para os casos de números (Int, Double, Float) e booleanos, temos que utilizar uma construção própria do Realm

```
let intOpcional      = RealmOptional<Int>()
let boolOpcional     = RealmOptional<Bool>()
let doubleOpcional   = RealmOptional<Double>()
let floatOpcional    = RealmOptional<Float>()
```

Type	Non-optional	Optional
Bool	@objc dynamic var value = false	let value = RealmOptional<Bool>()
Int	@objc dynamic var value = 0	let value = RealmOptional<Int>()
Float	@objc dynamic var value: Float = 0.0	let value = RealmOptional<Float>()
Double	@objc dynamic var value: Double = 0.0	let value = RealmOptional<Double>()
String	@objc dynamic var value = ""	@objc dynamic var value: String? = nil
Data	@objc dynamic var value = Data()	@objc dynamic var value: Data? = nil
Date	@objc dynamic var value = Date()	@objc dynamic var value: Date? = nil
Object	n/a: must be optional	@objc dynamic var value: Class?
List	let value = List<Type>()	n/a: must be non-optional
LinkingObjects	let value = LinkingObjects(fromType: Class.self, property: "property")	n/a: must be non-optional

USANDO O REALM

---

COMO  
CRIAR  
RELACIONAMENTOS?

# USANDO O REALM

## COMO CRIAR RELACIONAMENTOS?

- ▶ Nós já até vimos

```
@objc dynamic var objeto: ObjetoRealm?
```

- ▶ Temos como criar três tipos de relacionamento:

- ▶ Many-to-one

- ▶ Many-to-many

- ▶ Relacionamento inverso

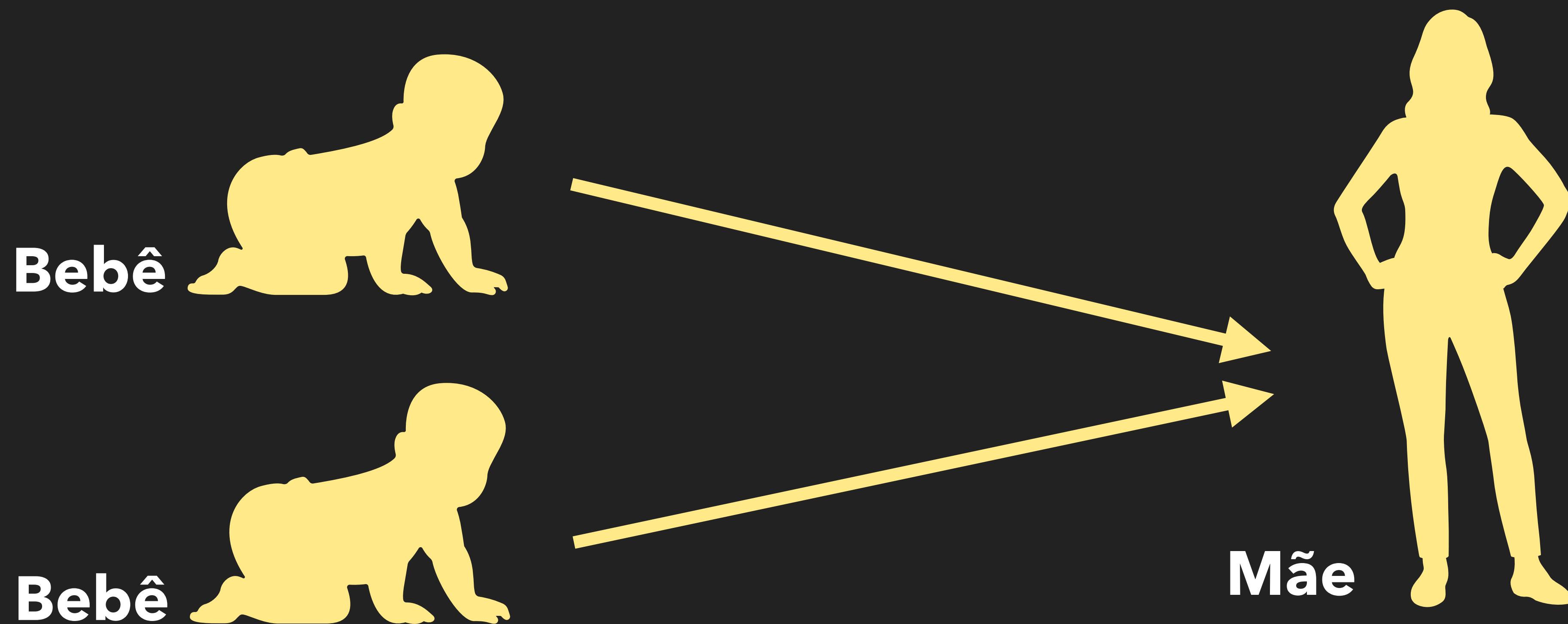
Object	n/a: must be optional	@objc dynamic var value: Class?
List	let value = List<Type>()	n/a: must be non-optional
LinkingObjects	let value = LinkingObjects(fromType: Class.self, property: "property")	n/a: must be non-optional

# USANDO O REALM

## MANY-TO-ONE

```
@objc dynamic var objeto: ObjetoRealm?
```

- ▶ Uma classe possui um elemento de outra
- ▶ Nada impede que mais de uma instância tenha o mesmo objeto como referência



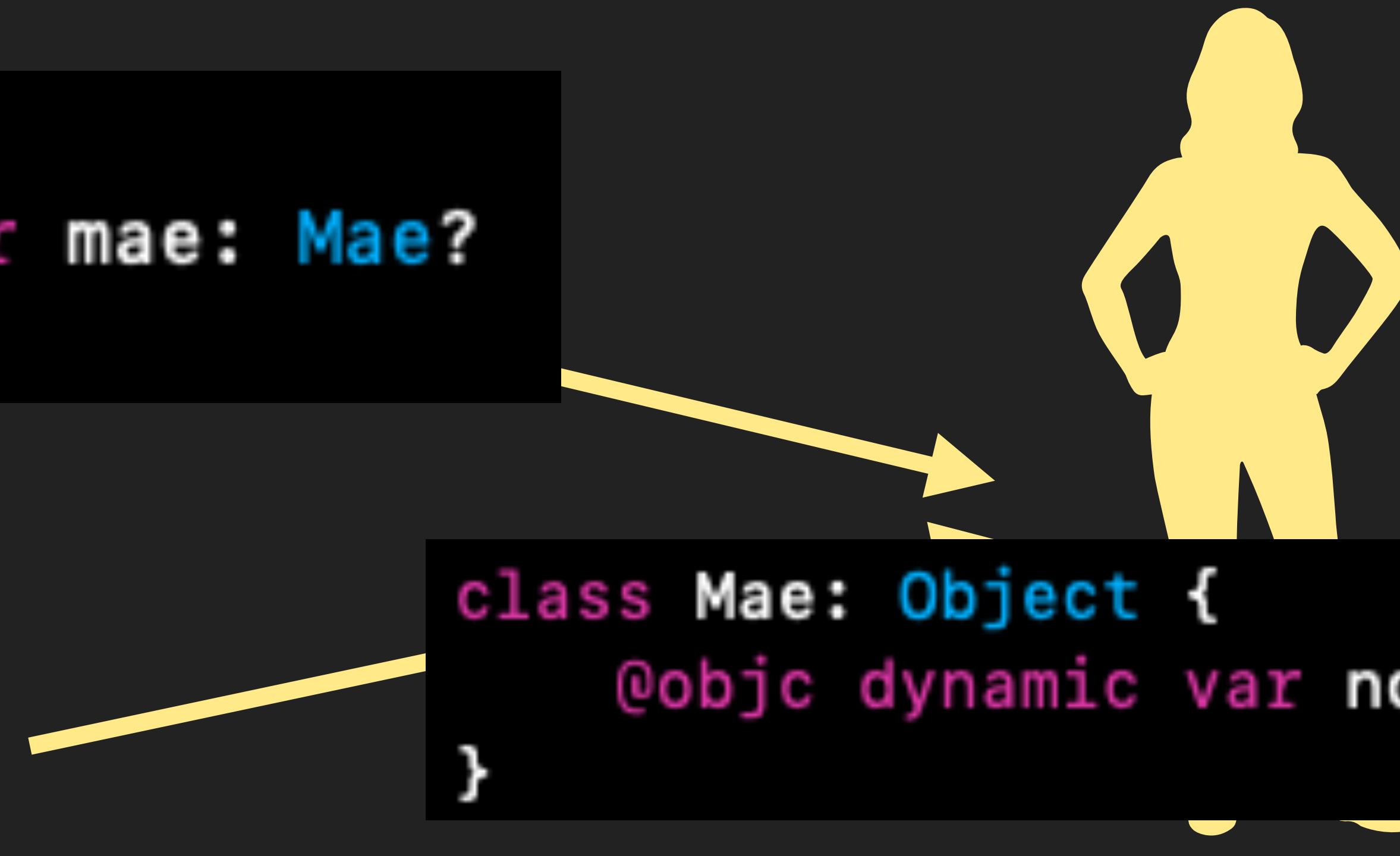
# USANDO O REALM

## MANY-TO-ONE

```
@objc dynamic var objeto: ObjetoRealm?
```

- ▶ Uma classe possui um elemento de outra
- ▶ Nada impede que mais de uma instância tenha o mesmo objeto como referência

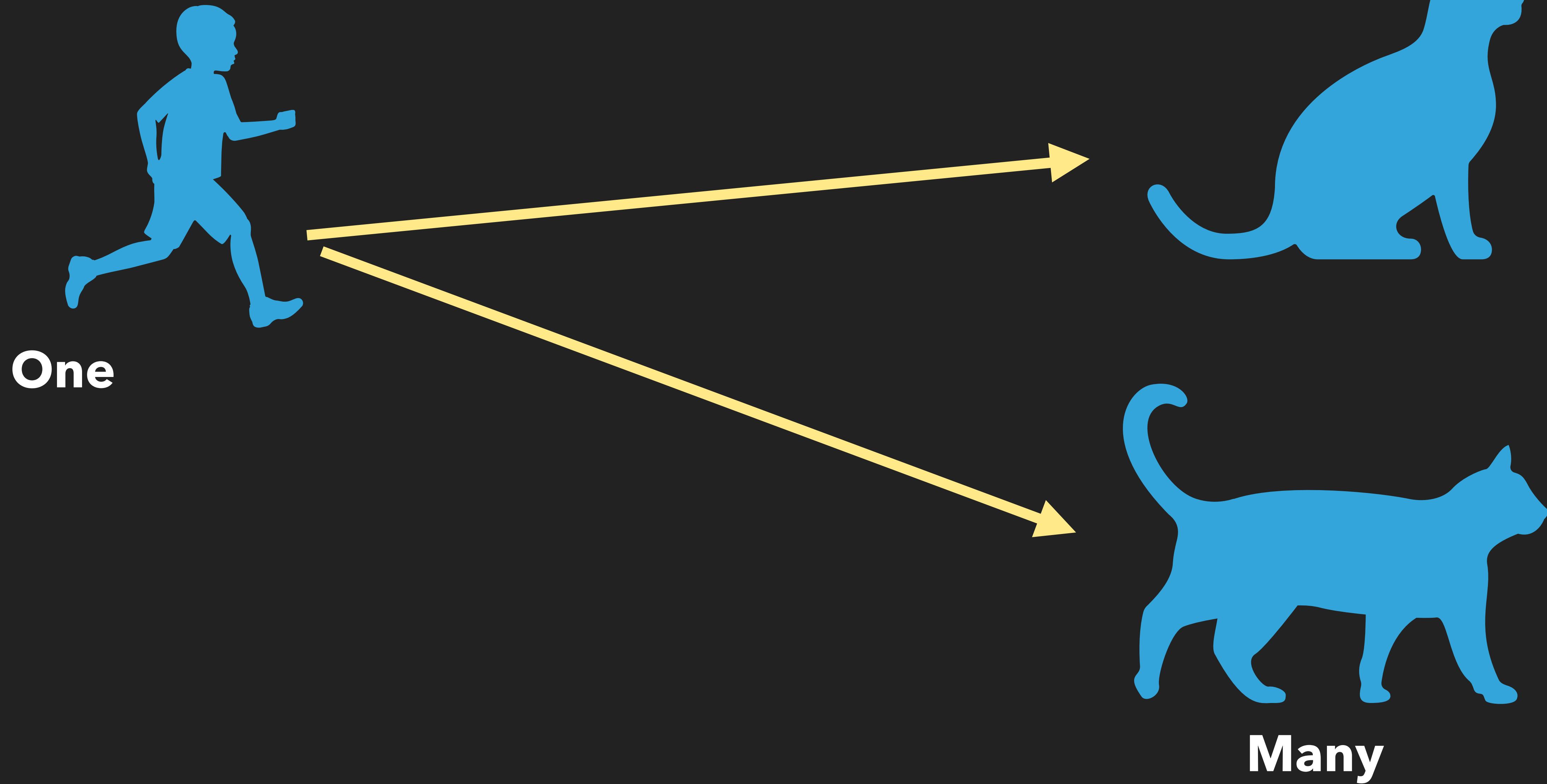
```
class Bebe: Object {  
    @objc dynamic var mae: Mae?  
}
```



Bebê

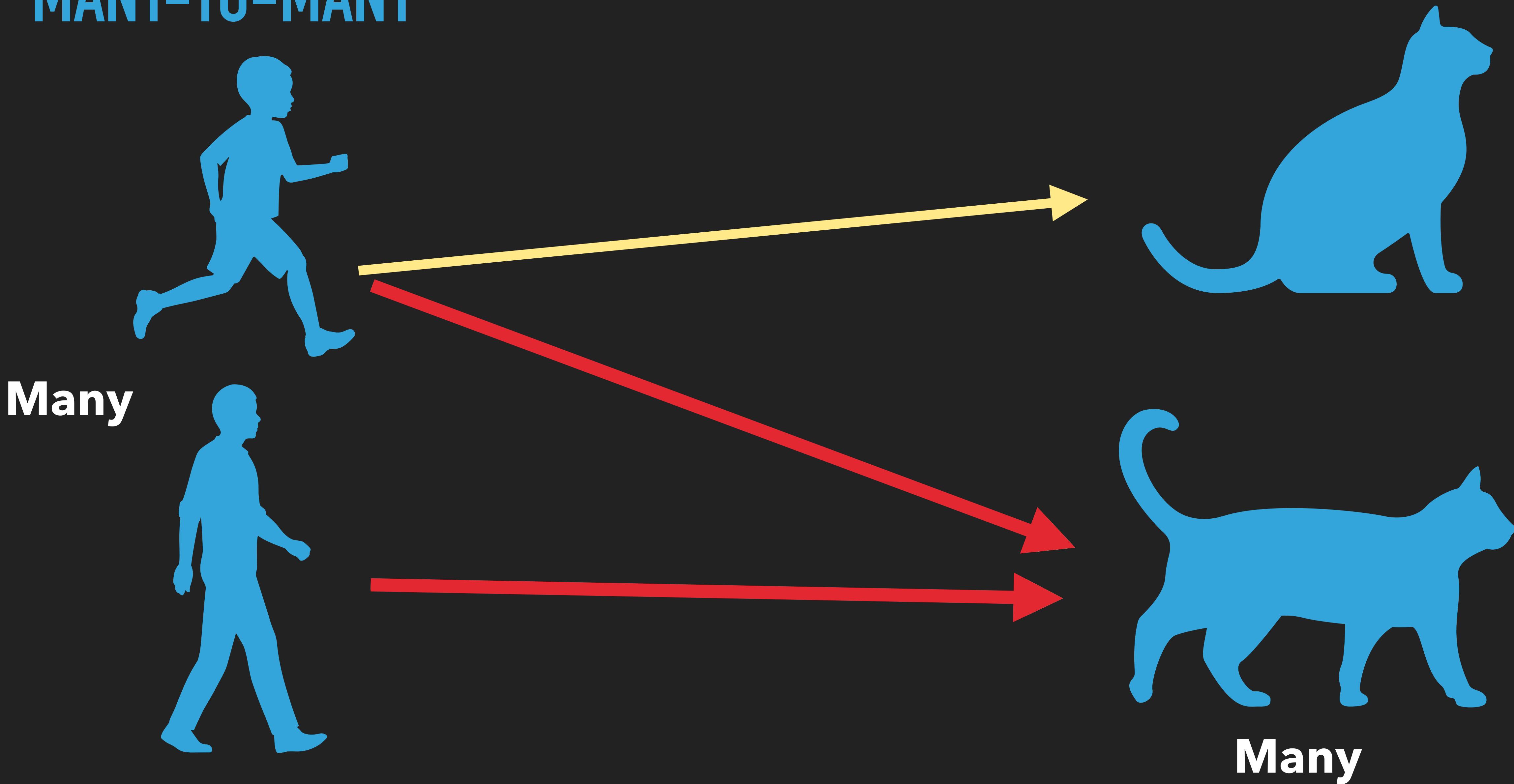
# USANDO O REALM

## MANY-TO-MANY



# USANDO O REALM

## MANY-TO-MANY



# USANDO O REALM

---

## MANY-TO-MANY

- ▶ Uma lista que pode ser do tipo de um objeto ou de qualquer um daqueles tipos suportados pelo Realm
- ▶ Pode conter o mesmo objeto mais de uma vez
- ▶ Preservam a ordem que os objetos foram inseridos

```
class Pessoa: Object {  
    let bichos = List<Animal>()  
}
```

# USANDO O REALM

## MANY-TO-MANY

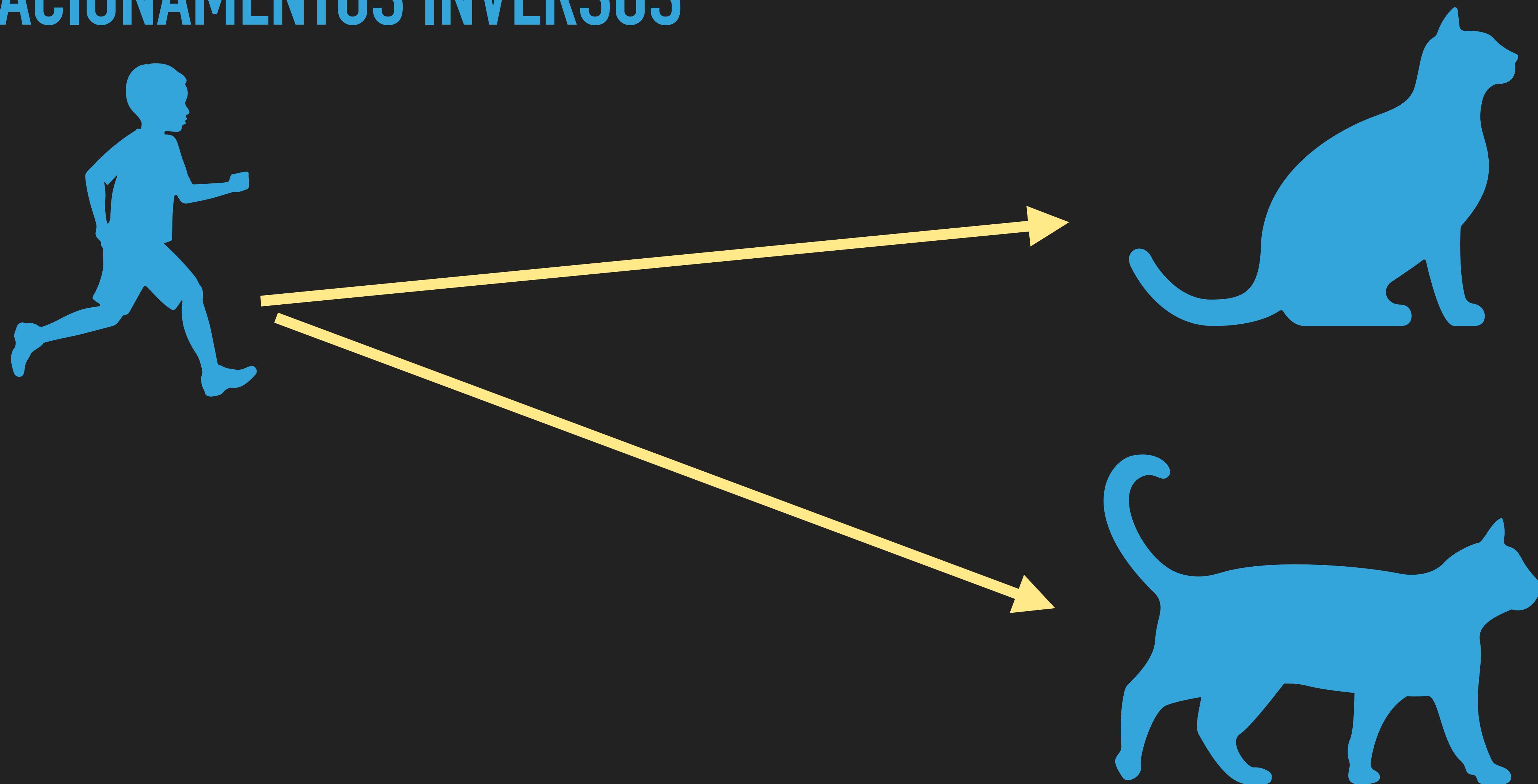
- ▶ Você acessa como uma lista

```
let bichosEstimacao = realm.objects(Animais.self).filter("nome contains 'Fred'"  
jose.bichos.append(objectsIn: bichosEstimacao)  
jose.bichos.append(rex)  
jose.bichos.append(rex)
```

- ▶ Atualmente, não é possível fazer um filtro de uma List com valores primitivos
  - ▶ Int, Float, Double, String, Data, Date

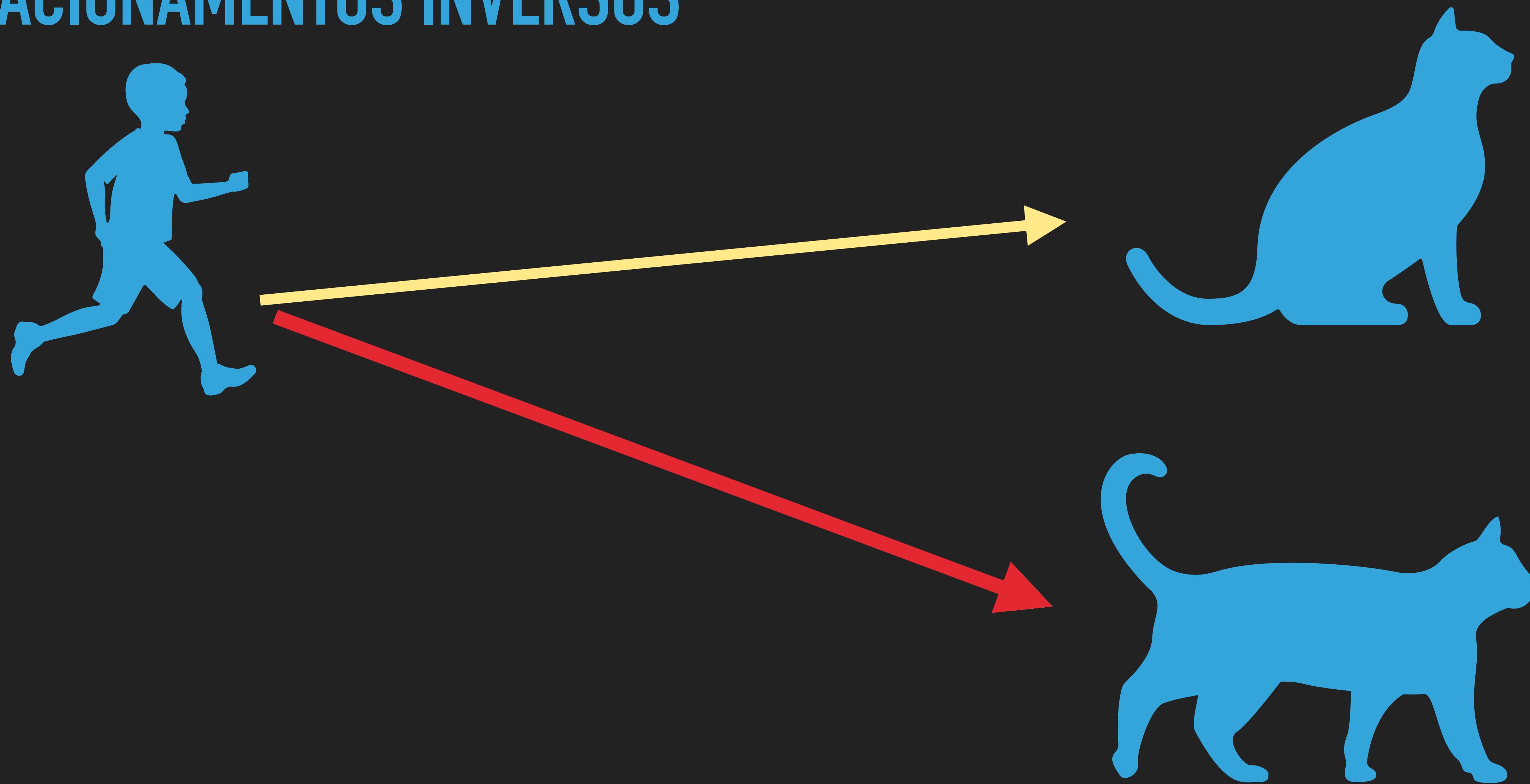
# USANDO O REALM

## RELACIONAMENTOS INVERSOS



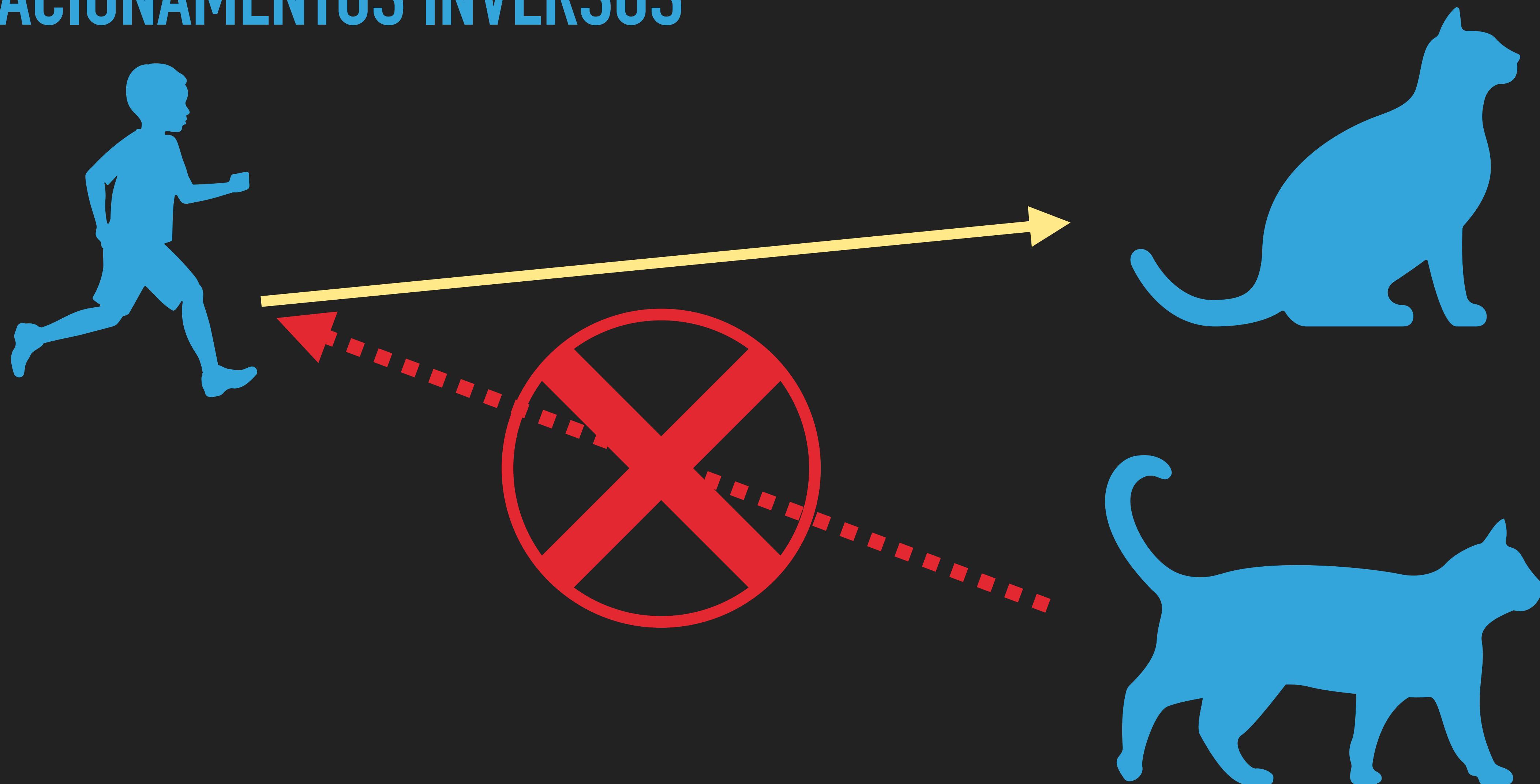
# USANDO O REALM

## RELACIONAMENTOS INVERSOS



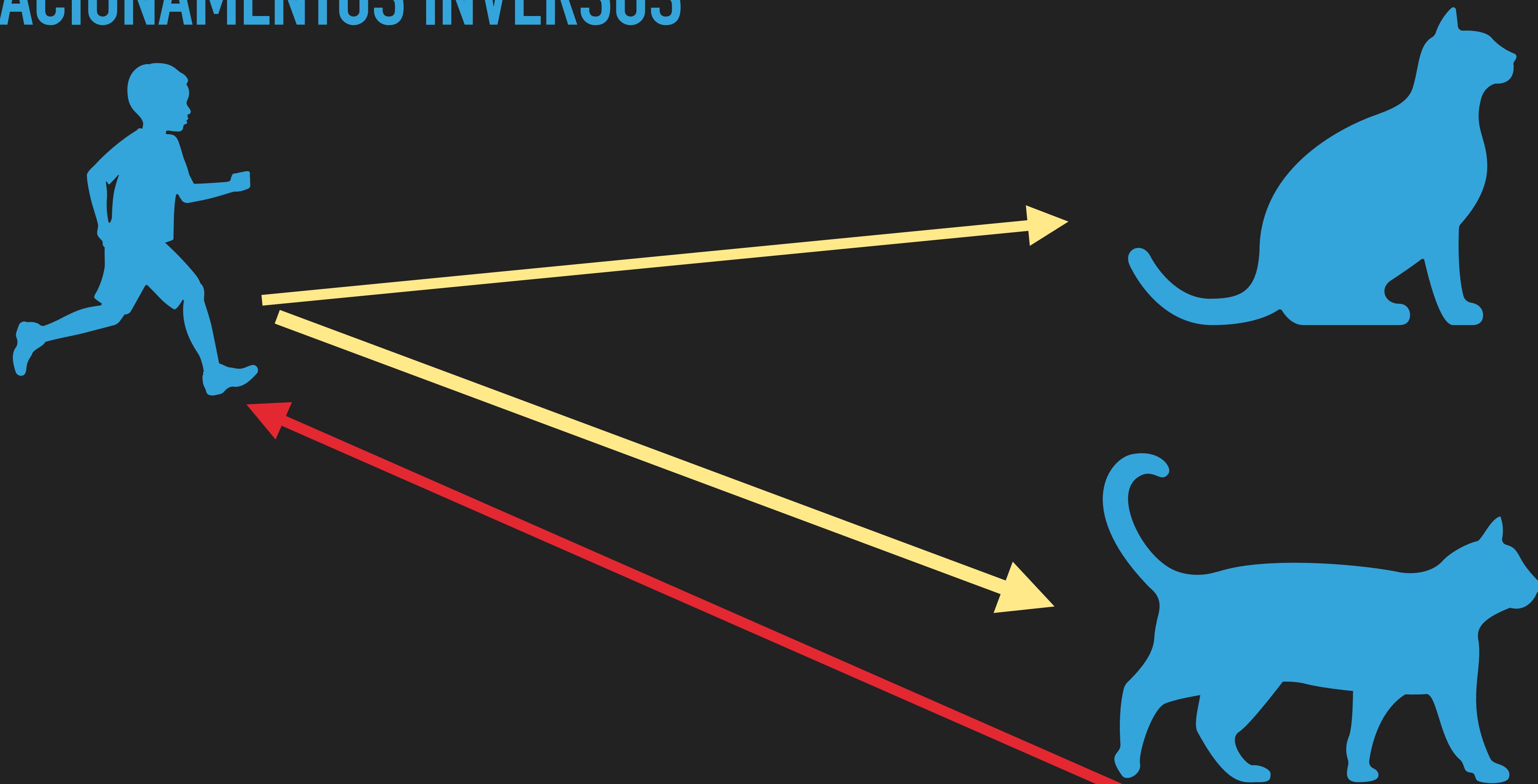
# USANDO O REALM

## RELACIONAMENTOS INVERSOS



# USANDO O REALM

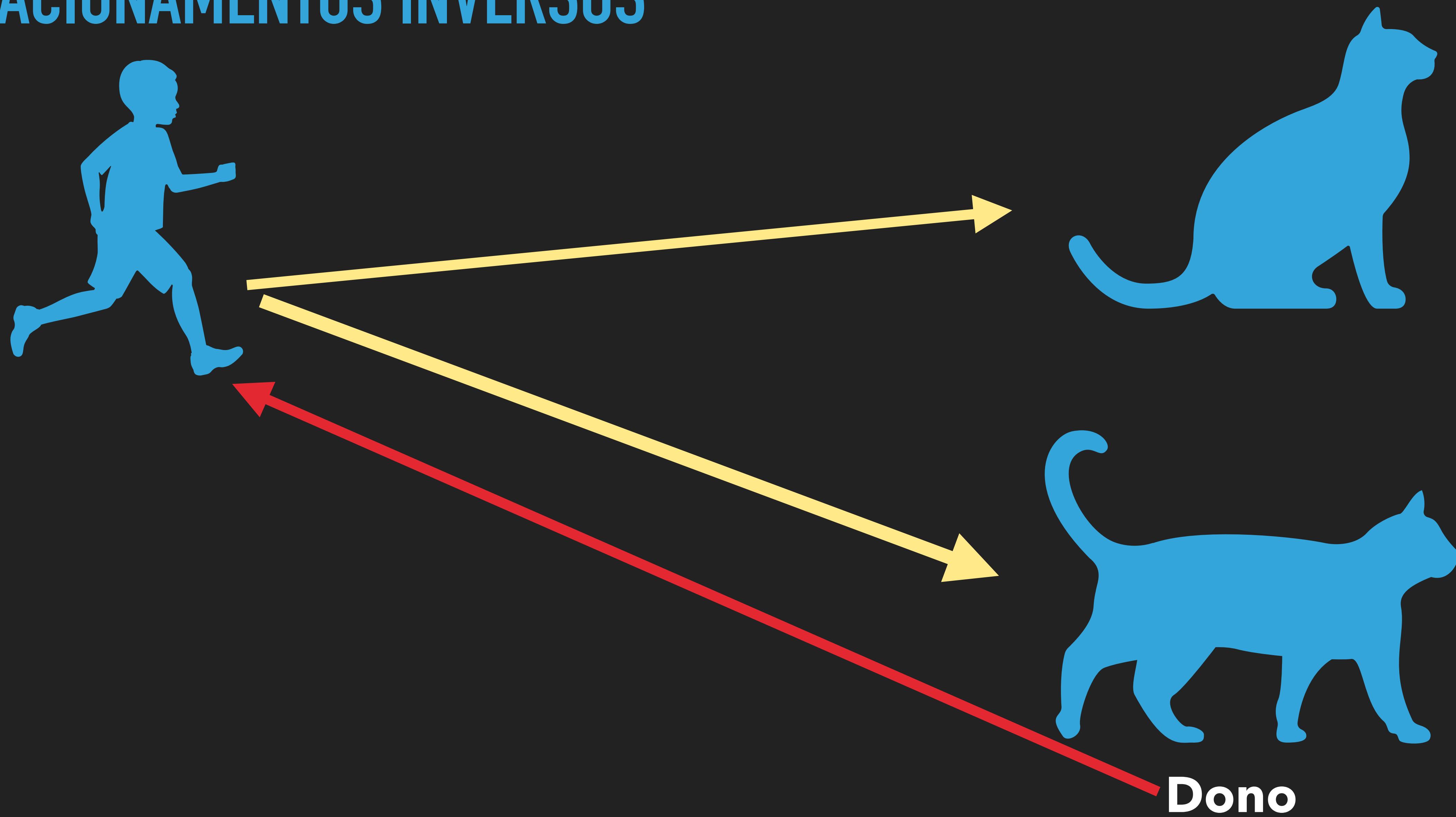
## RELACIONAMENTOS INVERSOS



@objc dynamic var dono: Pessoa?

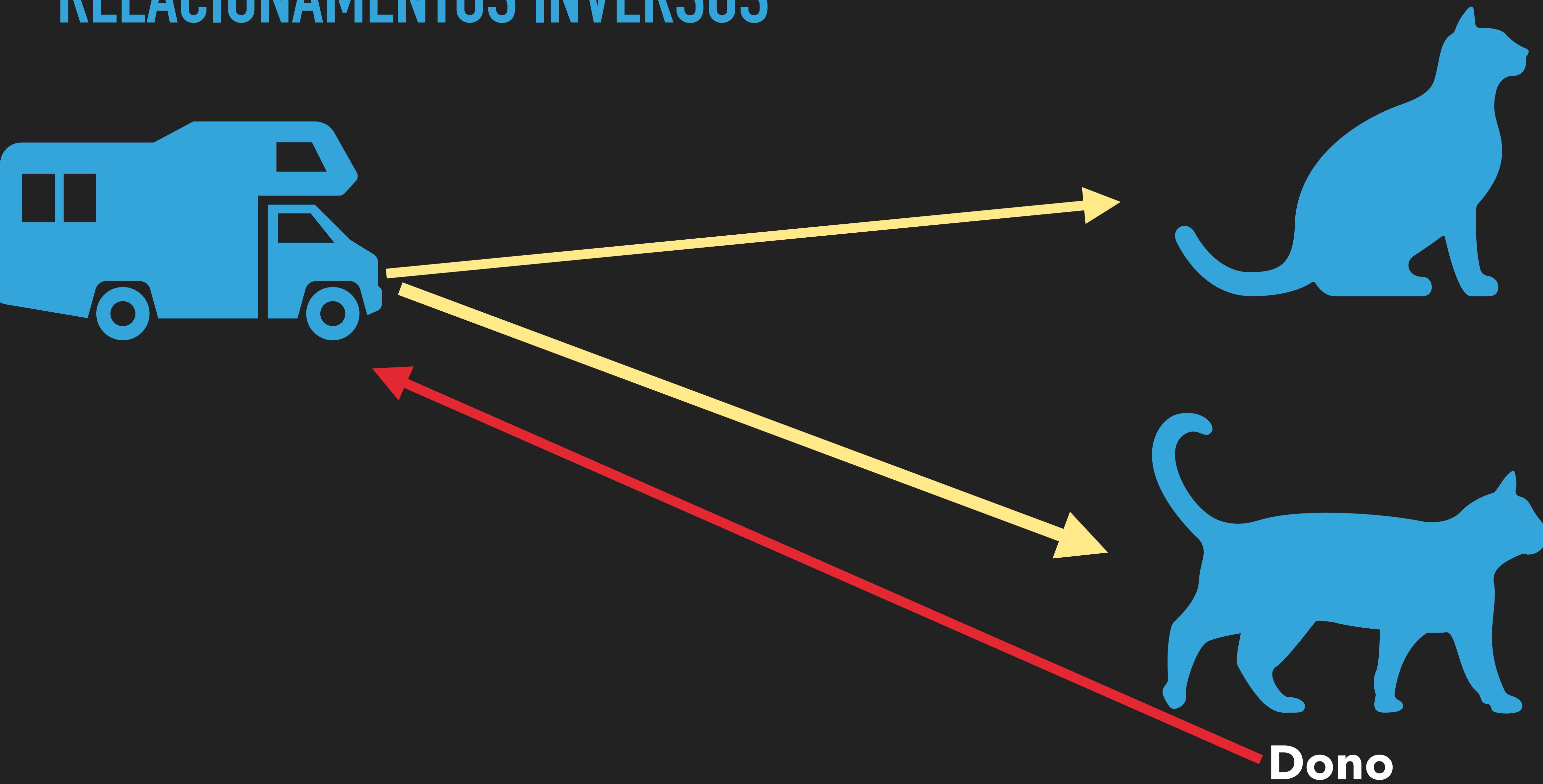
# USANDO O REALM

## RELACIONAMENTOS INVERSOS



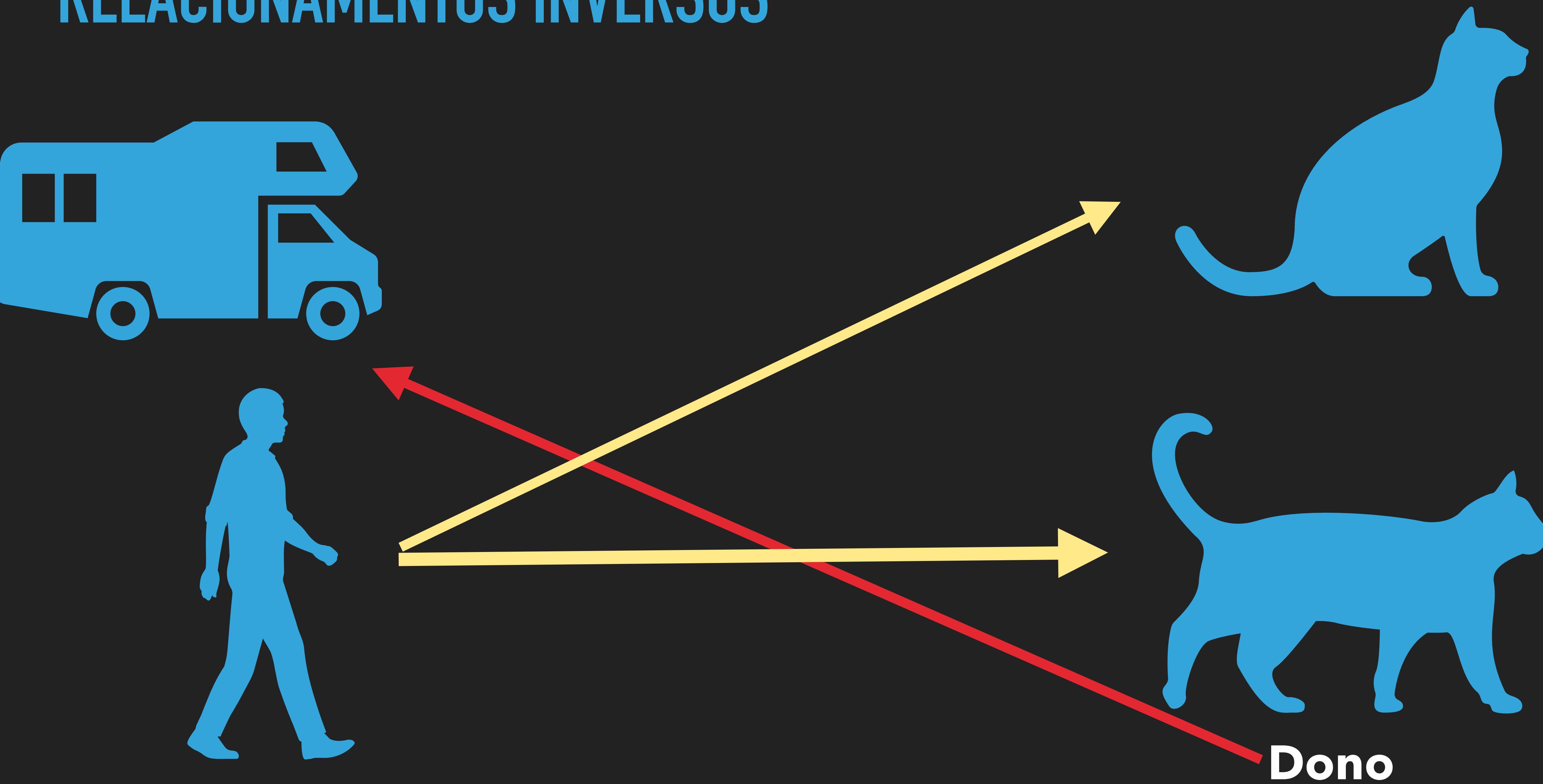
# USANDO O REALM

## RELACIONAMENTOS INVERSOS



# USANDO O REALM

## RELACIONAMENTOS INVERSOS



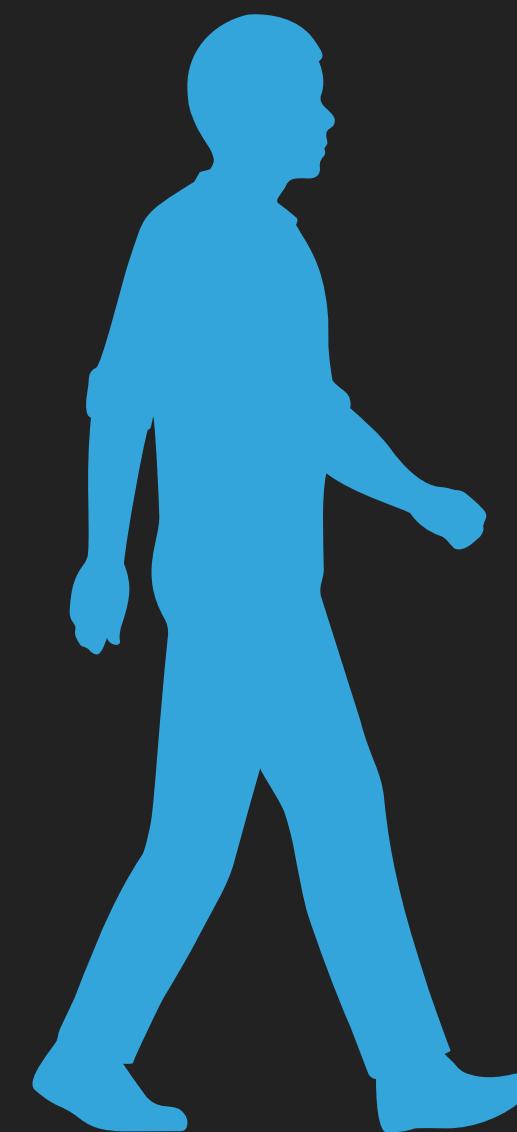
# USANDO O REALM

## RELACIONAMENTOS INVERSOS



# USANDO O REALM

## RELACIONAMENTOS INVERSOS



Relacionamentos  
são unidirecionais,  
essa é só uma  
representação visual



# USANDO O REALM

## RELACIONAMENTOS INVERSOS

- ▶ A 'ida' nós já criamos com a referência do objeto ou com a List na classe Pessoa que é dona de um animal

```
class Pessoa: Object {  
    let bichos = List<Animal>()  
}
```

- ▶ Agora falta o outro relacionamento, o relacionamento inverso

# USANDO O REALM

## RELACIONAMENTOS INVERSOS

- ▶ Utilizamos um LinkingObjects, assim ele sempre será atualizado

```
class Animal: Object {  
    @objc dynamic var nome: String = ""  
  
    let dono = LinkingObjects(fromType: Pessoa.self, property: "bichos")  
}
```

USANDO O REALM

---

# COMO CRIAR UMA PRIMARY KEY?

# USANDO O REALM

## CRIANDO UMA PRIMARY KEY

- Sobreescreva o método primaryKey da sua model e retorne o nome da property que será a Primary Key

```
class Usuario: Object {  
    @objc dynamic var nome: String = ""  
    @objc dynamic var idade: Int = 0  
  
    override static func primaryKey() -> String? {  
        return "nome"  
    }  
}
```

# USANDO O REALM

---

## IGNORANDO PROPERTIES

- ▶ Caso você não queira que uma property seja salva:
  - ▶ Sobreescreve o método ignoredProperties da sua model e retorne o nome das properties que serão ignoradas

```
class Usuario: Object {  
    @objc dynamic var nome: String = ""  
    @objc dynamic var idade: Int = 0  
  
    override static func ignoredProperties() -> [String] {  
        return ["idade"]  
    }  
}
```

USANDO O REALM

---

COMO  
SALVAR  
UM OBJETO?

# USANDO O REALM

---

## COMO SALVAR UM OBJETO?

- ▶ Temos que buscar a referência do Realm

```
let realm = try! Realm()
```

# USANDO O REALM

## COMO SALVAR UM OBJETO?

- ▶ Temos que buscar a referência do Realm

```
do {  
    let realm = try Realm()  
} catch let error {  
    print(error)  
}
```

# USANDO O REALM

## COMO SALVAR UM OBJETO?

- ▶ Então iniciamos nossa model e preenchemos ela

```
let realm = try! Realm()  
  
let user = Usuario()  
user.nome = "Paulo"  
user.idade = 30
```

# USANDO O REALM

## COMO SALVAR UM OBJETO?

- ▶ Temos três formas semelhantes de criar um objeto:

```
let user = Usuario()  
user.nome = "Paulo"  
user.idade = 30
```

```
// Cria a partir de um dicionario  
let outroUser = Usuario(value: ["nome" : "Paulo", "idade": 30])
```

```
// Cria a partir de um array  
let terceiroUser = Usuario(value: ["Paulo", 30])
```

# USANDO O REALM

## COMO SALVAR UM OBJETO?

- ▶ Por fim, só executamos um add

```
let realm = try! Realm()

let user = Usuario()
user.nome = "Paulo"
user.idade = 30

try! realm.write {
    realm.add(user)
}
```

# USANDO O REALM

## COMO SALVAR UM OBJETO?

- ▶ Ou ainda

```
let realm = try! Realm()
```

```
let user = Usuario()  
user.nome = "Paulo"  
user.idade = 30
```

```
try! realm.write {  
    realm.add(user)  
}
```

```
let user = Usuario()  
user.nome = "Paulo"  
user.idade = 30  
  
try! Realm().write {  
    realm.add(user)  
}
```

# USANDO O REALM

## E SE MINHA PROPERTY FOSSE REALMOPTIONAL? COMO EU A PREENCHO?

- ▶ Nesse caso é só preencher a property value da property do seu objeto
- ▶ Ao criar o seu objeto, a property RealmOptional inicia como nil

```
class Usuario: Object {  
    @objc dynamic var nome: String = ""  
    let idade = RealmOptional<Int>()  
}
```

```
user.idade.value = nil
```

```
user.idade.value = 30
```

# USANDO O REALM

## COMO EU CONSIGO CRIAR MEU OBJETO ASSIM?

```
let joao = Usuario(nome: "Joao", idade: 20)
```

- ▶ Para isso, é só criar um convenience init na nossa classe Usuario

```
convenience init(nome: String, idade: Int) {  
    self.init()  
    self.nome = nome  
    self.idade = idade  
}
```

# USANDO O REALM

---

## DEVO SEMPRE CRIAR UM TRY! REALM() ? VARIÁVEL DE CLASSE?

- ▶ Ao executar o **try! Realm** pela primeira vez em uma thread, o objeto é criado
- ▶ Ao chamar novamente na mesma thread, o retorna a mesma instância já criada
- ▶ O Realm toma conta disso para você.
  
- ▶ Você pode ter uma variável que armazena o Realm
- ▶ Você só precisa fazer isso uma vez por thread
  
- ▶ Ou pode fazer o try Realm() sempre que chamar
- ▶ Faça o que vai simplificar mais o código para o seu caso (recomendação do Realm)

USANDO O REALM

---

E PARA  
ATUALIZAR?

# USANDO O REALM

---

- Nesse caso a sua model necessariamente necessita ter uma Primary Key

```
let user = Usuario()
user.nome = "Paulo"
user.idade = 30

try! Realm().write {
    realm.add(user)
}

user.idade = 31

try! Realm().write {
    realm.add(user, update: true)
}
```

## USANDO O REALM

---

- ▶ Também é possível atualizar um objeto alterando sua property dentro de um write:

```
try! Realm().write {  
    user.idade = 31  
}
```

# USANDO O REALM

---

- ▶ Uma última forma de atualizar é passando o nome da propriedade e o valor a ser alterado. Isso é muito bom para:
  - ▶ Bulk editing
  - ▶ Alterações que você só sabe a propriedade a ser alterada em tempo de execução

```
let users = try! Realm().objects(Usuario.self)

try! Realm().write {
    users.setValue(31, forKeyPath: "idade")
}
```

USANDO O REALM

---

E PARA  
REMOVER?

# USANDO O REALM

- Com esse comando, você pode remover tanto um elemento quanto uma lista

```
try! Realm().write {  
    realm.delete(user)  
}
```

Void delete(object: Object)  
Void delete(objects: List<Element>)  
Void delete(objects: Results<Element>)  
Void delete(objects: Sequence)  
Void deleteAll()

USANDO O REALM

---

E PARA  
BUSCAR?

# USANDO O REALM

---

## QUERIES

- ▶ O Realm funciona de uma forma diferente do convencional. As queries são **lazy**.
- ▶ O Objeto não é carregado na memória, somente quando vamos acessar as properties dele.
- ▶ **Lazy** - É uma feature do Swift. É possível declarar variáveis assim.
- ▶ O resultado não é uma cópia do objeto, mas o objeto em sí
- ▶ A query mais simples nós já vimos:

```
let users = try! Realm().objects(Usuario.self)
```

# USANDO O REALM

## QUERIES

- ▶ Para filtrar nosso resultado, podemos usar uma string com a nossa query ou um predicate:

```
let filtrados = realm.objects(Usuario.self)
    .filter("idade > 29 AND nome BEGINSWITH 'Jo'")
```

```
let predicate = NSPredicate(format: "color = %@ AND nome BEGINSWITH %@", "tan", "Jo")
let filtrados2 = realm.objects(Usuario.self).filter(predicate)
```

# USANDO O REALM

---

## QUERIES

- ▶ ==, != - Aceitos por tudo até objetos e podem compararar com **nil**
- ▶ <= , <, >, >=, - Usados por Ints, Float, Double e Date
- ▶ BEGINSWITH, CONTAINS, ENDSWITH - String e Data - **CONTAINS 'casa'**
  - ▶ **CONTAINS[c] 'casa'** - Case-insensitive
  - ▶ **CONTAINS[d] 'casa'** - Diacritic-insensitive
  - ▶ **CONTAINS[cd] 'casa'** - Podem ser combinados
- ▶ LIKE - Strings
  - ▶ ? e \* - ? busca umchar e o \* busca zero ou mais
  - ▶ ?as\* - a - ca - asa - casa

# USANDO O REALM

## QUERIES

- ▶ AND, OR, NOT - Usados para unir filtros

```
.filter("nome CONTAINS[cd] 'jo' and idade > 29")
```

- ▶ ANY - Retorna um true/false indicando se aquela condição existe **ANY**  
**usuario.idade > 29**
- ▶ @count, @min, @max, @sum, @avg - funções que retornam um valor agregado

```
realm.objects(Usuario.self).filter("idade.@avg > 45")
```

# USANDO O REALM

---

## QUERIES

- ▶ Conseguimos ordenar utilizando o sorted e podemos indicar se é crescente ou decrescente:
- ▶ O padrão é ascendente

```
try! Realm().objects(Usuario.self).sorted(byKeyPath: "nome")
```

```
sorted(byKeyPath: "nome", ascending: true)
```

E A QUESTÃO  
DOS OBJETOS SE  
ATUALIZAREM?

## USANDO O REALM

---

### ATUALIZAÇÃO DOS OBJETOS

- ▶ O Object do Realm é um objeto que atualiza todas as referências a qualquer mudança.
- ▶ Isso deixa o código mais limpo
- ▶ Você pode trabalhar de uma forma que você é notificado a cada vez que certa Model se atualizar, assim, você pode atualizar a UI.
- ▶ Não vamos usar essas notificações em nosso App. Caso queira saber mais, leia sobre em:

[Realm Notifications](#)

- ▶ Queremos somente saber se os objetos atualizam automaticamente...

# USANDO O REALM

## ATUALIZAÇÃO DOS OBJETOS

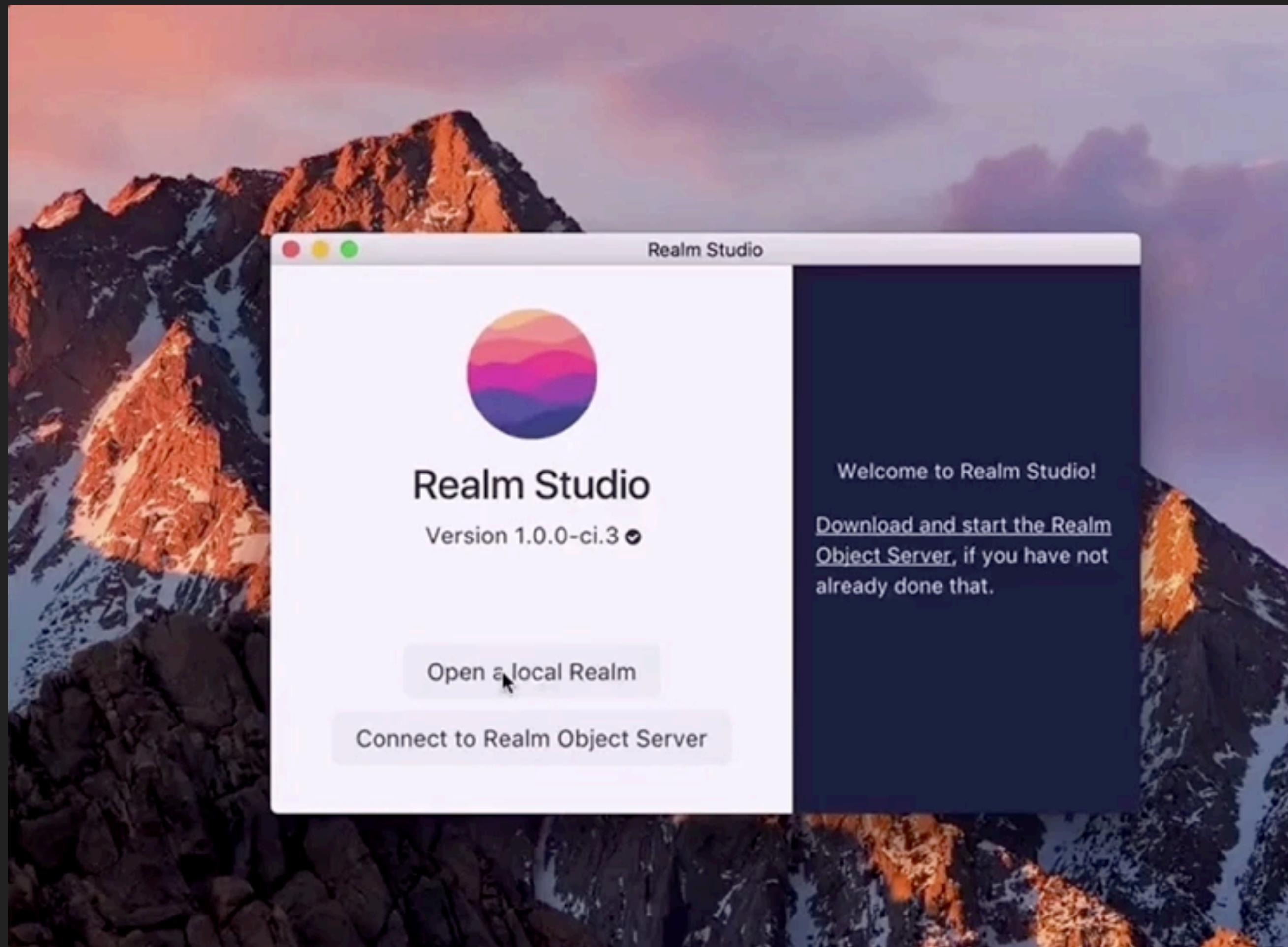
```
let myDog = Dog()  
myDog.name = "Fido"  
myDog.age = 1  
  
try! realm.write {  
    realm.add(myDog)  
}  
  
let myPuppy = realm.objects(Dog.self).filter("age == 1").first  
try! realm.write {  
    myPuppy!.age = 2  
}  
  
print("age of my dog: \(myDog.age)") // => 2
```

USANDO O REALM

---

COMO  
VER O MEU  
BANCO DE DADOS?

# USANDO O REALM REALM APP



## REALM STUDIO

# USANDO O REALM

---

## REALM APP

```
print("Realm location ")
print(Realm.Configuration.defaultConfiguration.fileURL)
```



---

# REALM - TÓPICOS AVANÇADOS

# REALM - TÓPICOS AVANÇADOS

## DICA - BULK OPERATIONS

- ▶ Bulk operations - operações em massa
- ▶ Tente executar o máximo possível dentro do **write**
  - ▶ Evita abrir e fechar conexões que deveriam ser mantidas abertas

```
for item in results {  
    try! realm.write {  
        item.value = newValue  
    }  
}
```



```
try! realm.write {  
    for item in results {  
        item.value = newValue  
    }  
}
```



# REALM - TÓPICOS AVANÇADOS

---

## DICA - PAGINAÇÃO

- ▶ Geralmente é implementado quando se tem uma base de dados muito grande
- ▶ A não ser que você vá copiar o Realm Results para um Array pela questão de filtro, você não deve fazer
  - ▶ Os objetos do Realm não são copiados todos para a memória diretamente. Eles são carregados a medida que suas properties vão sendo utilizadas.
  - ▶ Só é necessário que você limite a exibição pelo array que você já buscou, caso seja realmente necessário.
  - ▶ Lembre-se **lazy** load. Tente sempre tirar vantagem disso.

# REALM - TÓPICOS AVANÇADOS

---

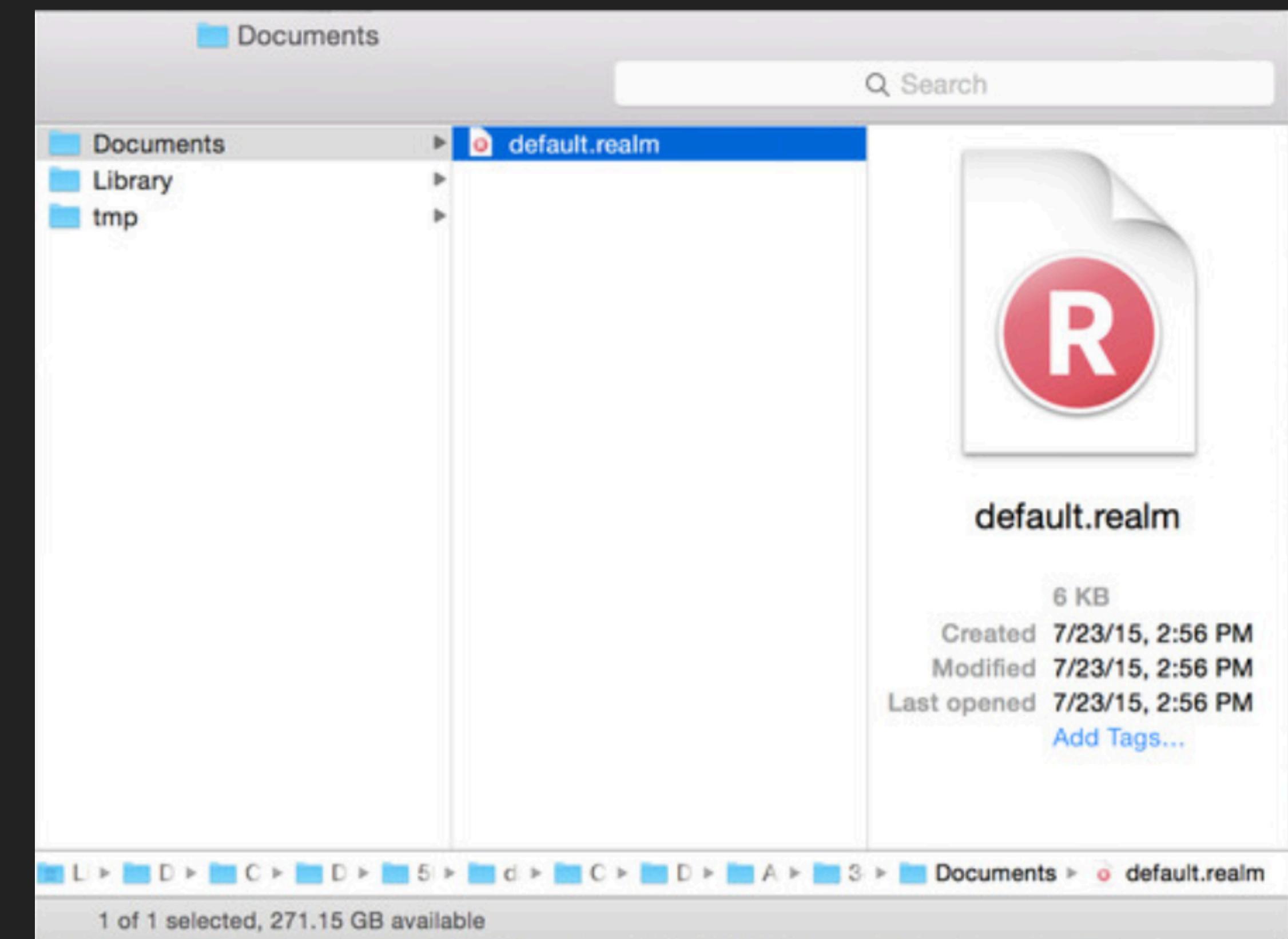
## TRABALHANDO COM THREADS

- ▶ Pode ler em threads diferentes
- ▶ Ao escrever, ele bloqueia as outras threads de escrita
  - ▶ Garantir que nenhum dado seja perdido
- ▶ Não é possível passar objetos de uma thread para outra. A boa prática recomendada pelo próprio Realm é:
  - ▶ Passar a primaryKey do objeto para a outra thread
  - ▶ Com a primaryKey, é melhor buscar o objeto nessa outra thread que ele já virá atualizado

# REALM - TÓPICOS AVANÇADOS

## REALM CONFIGURATION

- ▶ Mudar localização/nome do default.realm
  - ▶ No caso de várias contas, um arquivo Realm para cada uma delas
- ▶ Adicionar uma chave de criptografia
- ▶ Um Realm que funciona somente na memória
- ▶ Ou outra configuração



## REALM.CONFIGURATION

# REALM - TÓPICOS AVANÇADOS

---

## ADICIONANDO UM REALM COM DADOS AO SEU APP

- ▶ Não precisa adicionar os objetos programaticamente na primeira execução
- ▶ Avaliar questão de tamanho do App a ser baixado da loja

[HTTPS://REALM.IO/DOCS/SWIFT/LATEST/  
#BUNDLING-A-REALM](https://realm.io/docs/swift/latest/#bundling-a-realm)

# REALM - TÓPICOS AVANÇADOS

---

## ENCRYPTANDO COM REALM

- ▶ É simples, apenas uma configuração no Realm.Configuration
- ▶ Passar essa configuração ao chamar o **try Realm**

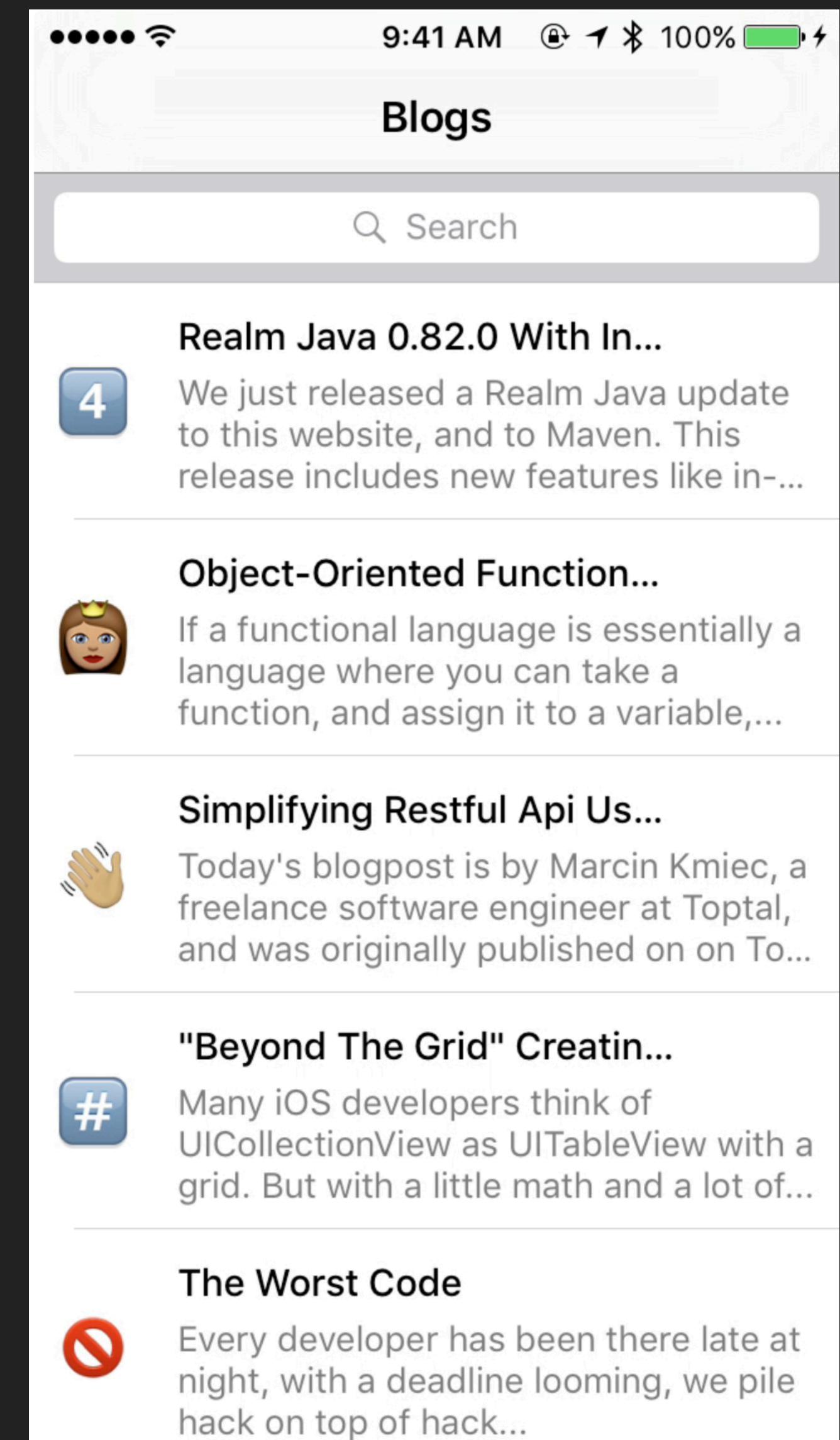
[HTTPS://ACADEMY.REALM.IO/POSTS/TIM-OLIVER-  
REALM-COCOA-TUTORIAL-ON-ENCRYPTION-WITH-  
REALM/](https://academy.realm.io/posts/tim-oliver-realms-cocoa-tutorial-on-encryption-with-realm/)

# REALM - TÓPICOS AVANÇADOS

## USAR A SEARCH CONTROLLER DO REALM

- ▶ O Realm criou um Pod que é uma Search Controller própria deles
- ▶ Possibilita determinar a busca no Storyboard
- ▶ Nada impede de adaptarmos usando a nossa

[HTTPS://ACADEMY.REALM.IO/](https://academy.realm.io/posts/building-an-ios-search-controller-in-swift/)  
[POSTS/BUILDING-AN-IOS-](#)  
[SEARCH-CONTROLLER-IN-SWIFT/](#)

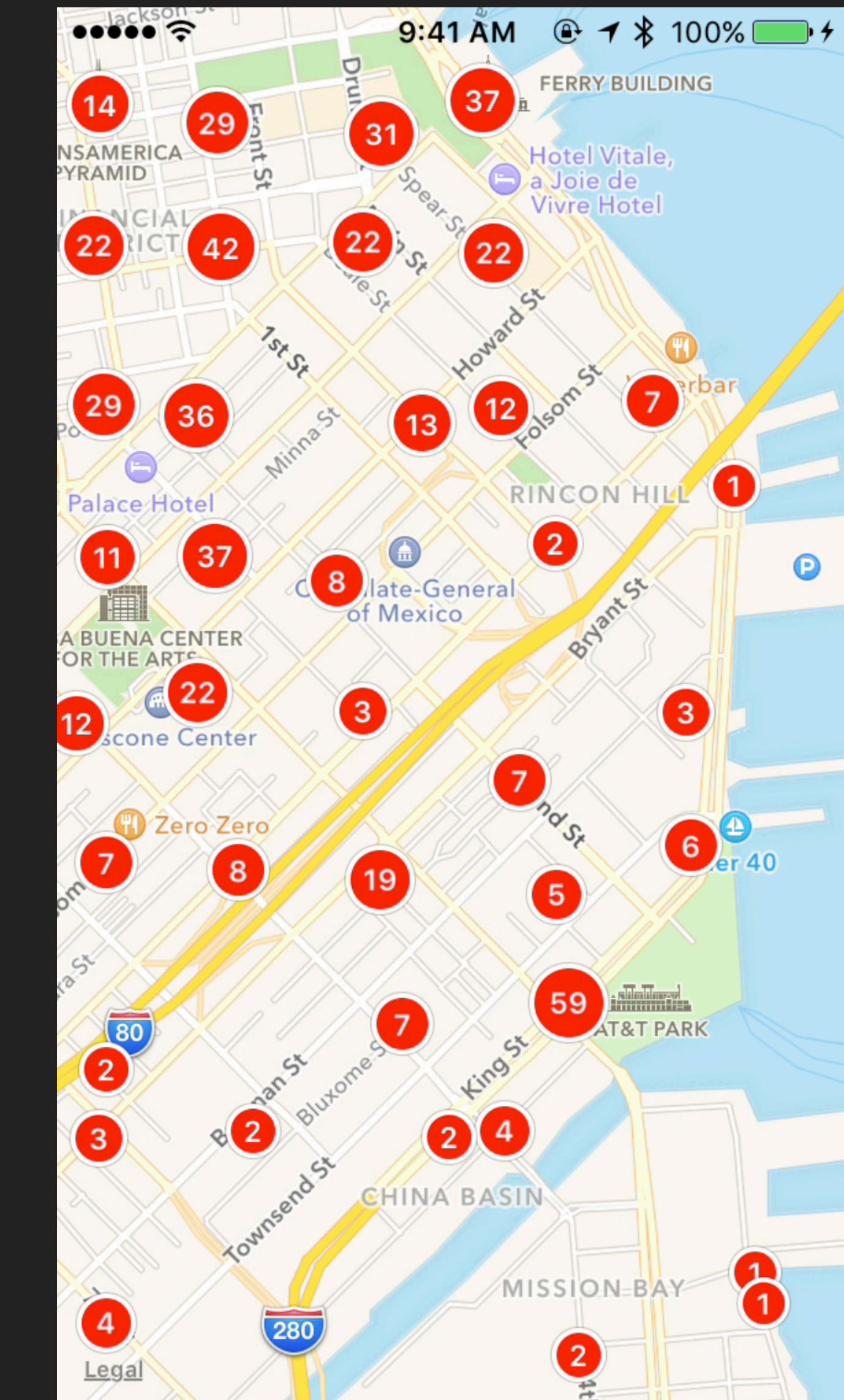


# REALM - TÓPICOS AVANÇADOS

## CLUSTER MAP DO REALM

- ▶ O Realm criou um Pod que é um Mapa próprio deles
- ▶ Agrupa os pontos próximos
- ▶ A configuração pode ser feita via Storyboard
- ▶ Nada impede de adaptarmos usando a nossa

[HTTPS://ACADEMY.REALM.IO/  
POSTS/BUILDING-AN-IOS-  
CLUSTERED-MAP-VIEW-IN-SWIFT/](https://academy.realm.io/posts/building-an-ios-clustered-map-view-in-swift/)



# REALM - TÓPICOS AVANÇADOS

---

## LIMITAÇÕES E SOLUÇÕES

- ▶ É sempre bom ver a documentação para qualquer limitação
- ▶ Melhor do que quebrar a cabeça
- ▶ Algumas delas envolvem tamanho máximo de caracteres para nome de classe e properties
- ▶ Outras quanto a tamanho máximo suportado para armazenar medias em formato Data

**CURRENT LIMITATIONS**