
citi

Hilton **Pintor**

Desenvolvedor (iOS/tvOS/watchOS)

hiltonpintor@gmail.com

citi

EMENTA PRELIMINAR DO CURSO:

SWIFT	STORYBOARD
Variáveis, constantes e operadores	Views
Tipos de variáveis (números, strings, entre outras)	InputView TextField
Coleções (array, dicionário)	ScrollView
Controle de fluxo (condicionais e loops)	TableView
Funções e closures	Navigation Controller
Enums	Tab Bar
Classes e structs	Labels
Protocols	Botões
Casting de tipos	Trocando de telas
Optionals Delegation	Integrando o storyboard com o código (outlets, buttons, entre outros)
Persistência	Webviews

// semana 02

Segunda	Terça	Quarta	Quinta	Sexta
~~TableView~~	~~Table View~~	~~Persistência~~	Projeto	Projeto
~~ScrollView~~				Extra

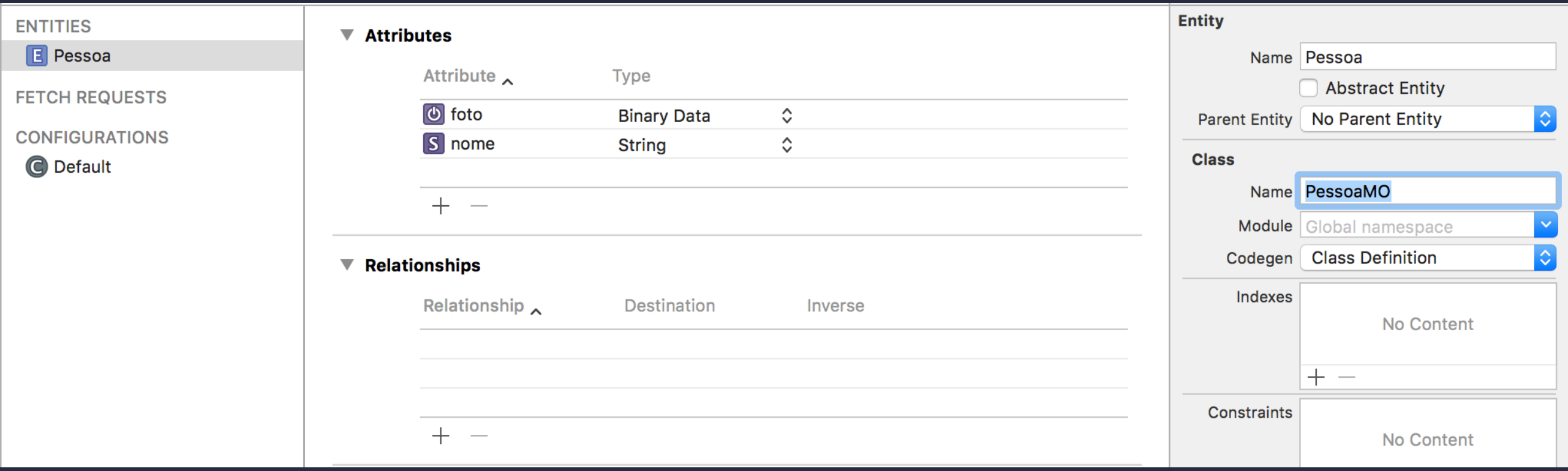
// Aula 09

// Dúvidas da Aula 08

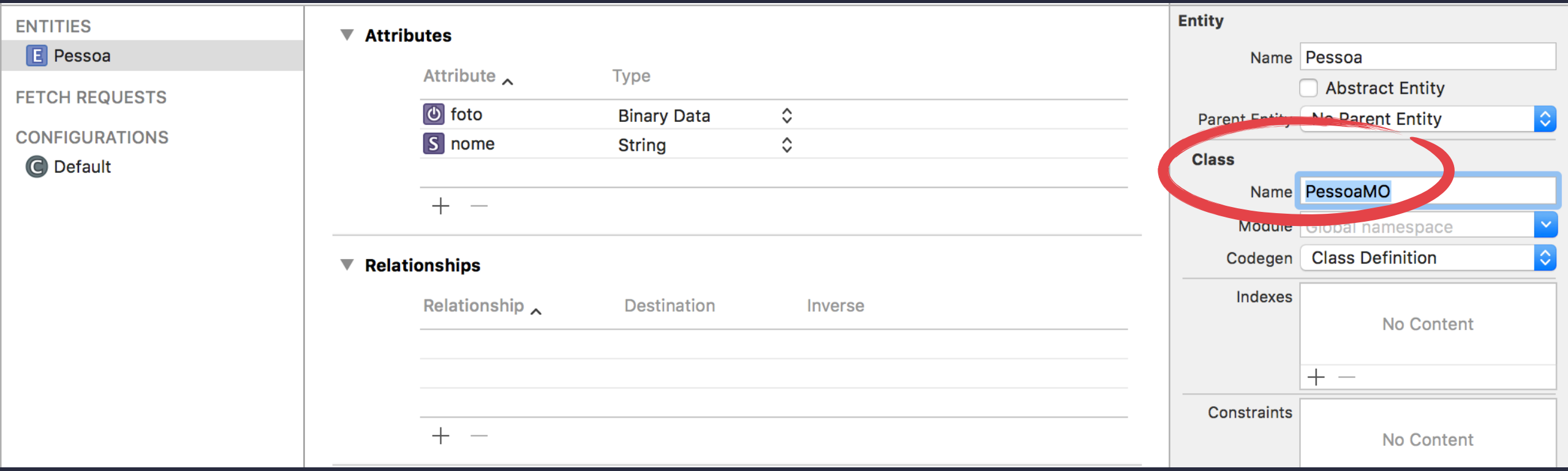
/*
Como usar Classes com Core Data
*/

citi

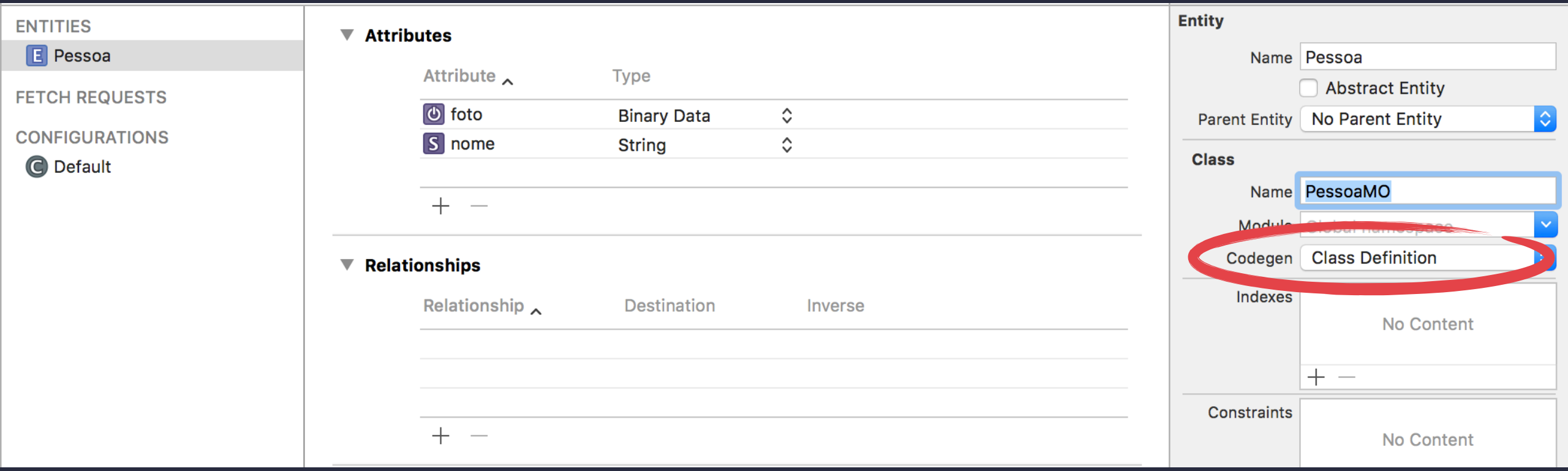
// Nome de Classe x Entidade



// Nome de Classe x Entidade



// Geração de Código



// Tipos de CodeGen

Class Definition:

1. opção "**plug and play**"
2. **não** permite gerar/editar os arquivos .swift
3. override e novas funcionalidades: **Extension**

// Tipos de CodeGen

Manual / None:

1. similar a primeira opção
2. **permite** gerar/editar os arquivos .swift
3. override e novas funcionalidades: **Extension**

// Tipos de CodeGen

Category / Extension:

1. propriedades **fora** do Core Data
2. arquivos criados na mão
3. override e novas funcionalidades: **Class** e **Extension**

/*
Realizar **fetch** dos dados salvos
*/

// 4. fetch **antigo**

```
class ViewController: UIViewController {
    var appDelegate: AppDelegate?
    var managedContext: NSManagedObjectContext?
    var pessoas: [NSManagedObject] = []

    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)

        let fetchRequest = NSFetchRequest<NSManagedObject>(entityName: "Pessoa")

        do {
            try self.pessoas = (self.managedContext?.fetch(fetchRequest))!
        } catch let error as NSError {
            print("erro na hora de pedir. \(error), \(error.userInfo)")
        }
    }
}
```

// 4. fetch **novo**

```
class ViewController: UIViewController {
    var appDelegate: AppDelegate?
    var managedContext: NSManagedObjectContext?
    var pessoas: [PessoaMO] = []

    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)

        let fetchRequest: NSFetchRequest<PessoaMO> = PessoaMO.fetchRequest()

       do {
            try self.pessoas = (self.managedContext?.fetch(fetchRequest))!
            self.tableView.reloadData()
        } catch let error as NSError {
            print("erro na hora de pedir. \(error), \(error.userInfo)")
        }
    }
}
```

/*
5. Quando adicionar novos dados, **save**
*/

// 5. save **antigo**

```
func save(novoNome: String) {
    let entity = NSEntityDescription.entity(forEntityName: "Pessoa", in: managedContext!)

    let pessoa = NSManagedObject(entity: entity!, insertInto: managedContext)

    let img = #imageLiteral(resourceName: "diego")
    let imgData = UIImageJPEGRepresentation(img, 1)

    pessoa.setValue(imgData, forKey: "foto")

    pessoa.setValue(novoNome, forKey: "nome")

    do {
        try managedContext?.save()
        self.pessoas.append(pessoa)
    } catch let error as NSError {
        print("erro na hora de salvar. \(error), \(error.userInfo)")
    }
}
```

// 5. save **novo**

```
func save(novoNome: String) {

    let novaPessoa = PessoaMO(context: self.managedContext)
    novaPessoa.foto = NSData(data: UIImageJPEGRepresentation(UIImage(named: "diego"), 1)!)
    novaPessoa.nome = novoNome

    do {
        try managedContext?.save()
        self.pessoas.append(pessoa)
        self.tableView.reloadData()

    } catch let error as NSError {
        print("erro na hora de salvar. \(error), \(error.userInfo)")
    }
}
```

/*

5. Quando remover dados, **delete**

*/

```swift
extension ViewController: UITableViewDataSource {
// 5. delete antigo
    func tableView(_ tableView: UITableView,
                   commit editingStyle: UITableViewCellEditingStyle,
                   forRowAt indexPath: IndexPath) {

        if editingStyle == .delete {
            let pessoa = self.pessoas[indexPath.row]
            self.managedContext?.delete(pessoa)
            self.appDelegate?.saveContext()

            let fetchRequest = NSFetchRequest<NSManagedObject>(entityName: "Pessoa")

            do {
                try self.pessoas = (self.managedContext?.fetch(fetchRequest))!
                tableView.reloadData()
            } catch {
                print("Fetching Failed")
            }
        }
    }
}
```

```swift
extension ViewController: UITableViewDataSource {
// 5. delete novo
    func tableView(_ tableView: UITableView,
                   commit editingStyle: UITableViewCellEditingStyle,
                   forRowAt indexPath: IndexPath) {

        if editingStyle == .delete {
            let pessoa = self.pessoas[indexPath.row]
            self.managedContext?.delete(pessoa)
            self.appDelegate?.saveContext()

            let fetchRequest: NSFetchRequest<PessoaMO> = PessoaMO.fetchRequest()

            do {
                try self.pessoas = (self.managedContext?.fetch(fetchRequest))!
                tableView.reloadData()
            } catch {
                print("Fetching Failed")
            }
        }
    }
}
```

// Resolução do Exercício 14

// Projeto

// Fizz Buzz - jogo/teste

// Fizz Buzz - jogo/teste

Escreva um programa que imprima números de **1 a 100**:

// Fizz Buzz - jogo/teste

Escreva um programa que imprima números de **1 a 100**:

// Fizz Buzz - jogo/teste

Escreva um programa que imprima números de **1 a 100**:

- Mas para múltiplos de **3**, imprima “**Fizz**" ao invés do número.

// Fizz Buzz - jogo/teste

Escreva um programa que imprima números de **1 a 100**:

- Mas para múltiplos de **3**, imprima “**Fizz**" ao invés do número.
- Para múltiplos de **5**, imprima “**Buzz**" ao invés do número.

// Fizz Buzz - jogo/teste

Escreva um programa que imprima números de **1 a 100**:

- Mas para múltiplos de **3**, imprima “**Fizz**" ao invés do número.
- Para múltiplos de **5**, imprima “**Buzz**" ao invés do número.
- E para números que são múltiplos de **3 e 5**, imprima “**FizzBuzz**"

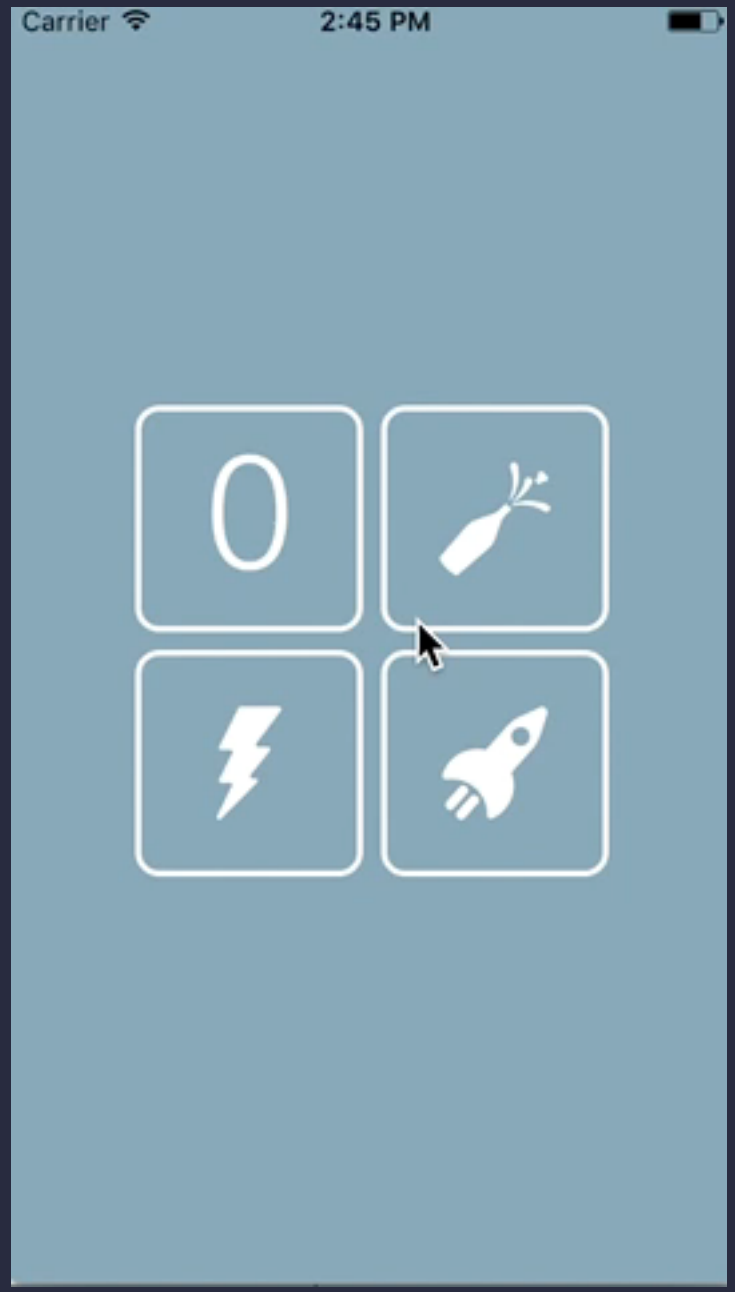
// Fizz Buzz - jogo/teste

Escreva um programa que imprima números de **1 a 100**:

- Mas para múltiplos de **3**, imprima “**Fizz**" ao invés do número.
- Para múltiplos de **5**, imprima “**Buzz**" ao invés do número.
- E para números que são múltiplos de **3 e 5**, imprima “**FizzBuzz**"

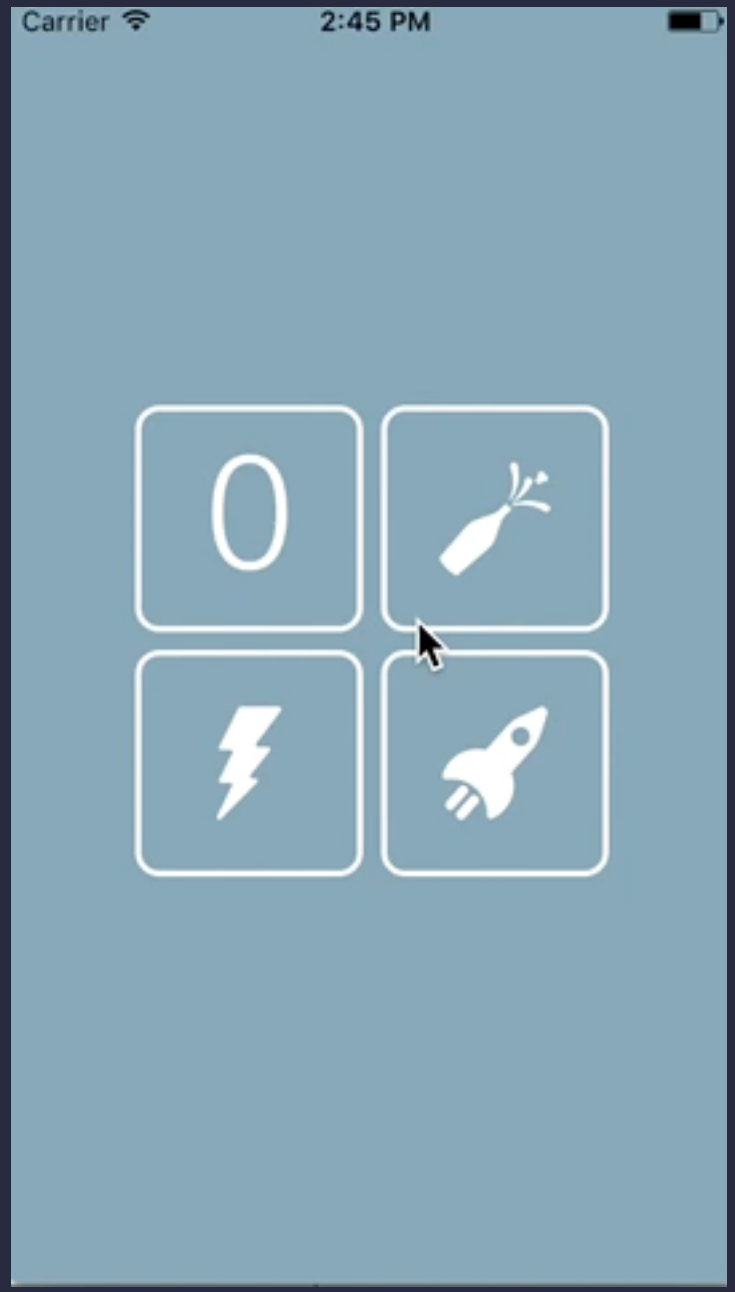
1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14, Fizz Buzz, 16, 17, Fizz, 19, Buzz, Fizz, 22, 23, Fizz, Buzz, 26, Fizz, 28, 29, Fizz Buzz, 31, 32, Fizz, 34, Buzz, Fizz, ...

// Fizz Buzz - App



interface meramente ilustrativa

// Fizz Buzz - App



interface meramente ilustrativa

// Tela de Jogo

Faça um app que permita que o jogador jogue uma partida de **fizz buzz** através do uso de botões da interface de seu app.

// Tela de Jogo

Faça um app que permita que o jogador jogue uma partida de **fizz buzz** através do uso de botões da interface de seu app.

Quando o usuário **errar** o valor da vez, ou o jogo **acabar**:

- o usuário deve ser direcionado para uma tela **de cadastro**

// Tela de Cadastro

Nessa tela o usuário pode:

- inserir seu **nome** (num text field)
- submeter sua **foto** (que deve ser mostrada em uma Image View)
- ver o seu **score** *número de acertos* (numa label)
- **salvar** os valores cadastrados (direcionando para **tela de scores**)
- **cancelar** (voltando para **tela de jogo** para **jogar novamente**)

// Tela de Scores

Nessa tela o usuário pode:

- Ver uma Table View onde cada célula tem:
  - **Nome** do jogador
  - **Foto** do jogador
  - **Score** obtido
- **Deletar** células da Table View
- Observar que o conteúdo da Table View tem **persistência**
- **Voltar** para tela de jogo sem passar pela de cadastro (unwind segue)

// Extra

- Ver os scores ordenados do maior pro menor

// Avaliação

// Domínio dos Conteúdos

Jornada de Cursos - CITi para mim, Walber 28 de jul

1. É obrigatória a realização de uma atividade de avaliação? Sim.
2. Quem não conseguir desempenho satisfatório na avaliação deixará de ganhar alguma coisa, como certificado? Para ter direito a certificado, o participante deve comparecer a, no mínimo, 7 aulas e ter 75% de aproveitamento na avaliação.

// Domínio dos Conteúdos

- programação em Swift
- MVC
- StoryBoard
- Botões
  - IBActions
- Labels
- Text Fields
- Image View
- Navigation Controller
- Segues (show)
- Segues (unwind)
- Pedir permissão de privacidade
- Table View
  - Células customizadas
  - Data Source
  - Interação com Células
    - seleção
    - inserção
    - remoção
- Persistência (Core Data)

// Domínio dos Conteúdos

- **Quinta** e **Sexta** serão utilizados para a avaliação
- É permitido:
  - Consultas online
  - Consultas offline
  - Tirar dúvidas com instrutor
  - Tirar dúvidas com colegas
  - Fazer projeto em casa
- É proibido:
  - entregar projeto feito por terceiros

// Submissão

- Nome Completo
- zip/rar do projeto

hiltonpintor@gmail.com


DÚVIDAS
?