05

Reaproveitamento de Célula

Reaproveitamento de célula

Description

Pelo `NSLog` nós indicamos cada campo que queremos que aparecesse: nome, e-mail, endereço, etc. Seria muito mais interessante se só pedíssemos para adicionar um contato e o programa pegasse as informações.

Em "Contato.m" fazemos:

```
@implementation Contato

- (NSSTring *)description {
    NSString *dados = [NSString stringWithFormat:@"Nome: %@ Endereço: %@ E-mail: %@ Site: %@ Telefor
    return dados;

}

@end
```

E no ViewController:

```
-(void) adiciona {
    //...

    NSLog(@"%@", contato);

}
```

Ao adicionarmos um contato novo, no log aparece:

Poderíamos chamar a `description` explicitamente, mas não é necessário:

```
NSLog(@"%@", [contato description]);
```

NSMutablearray

Precisamos criar uma estrutura que seja capaz de armazenar mais de um contato de uma só vez. Usaremos a estrutura de *array* para isso com a Classe "NSArray" no ViewController dentro do método `adiciona` :

```
NSArray *array = [NSArray new];
```

O comportamento dessa classe é imutável, ou seja, não conseguimos adicionar mais contatos além da quantidade de objetos que indicarmos. O que não faz sentido, pois não sabemos a quantidade de contatos que teremos numa lista. Logo usaremos a Classe "NSMutableArray":

```
NSMutableArray *contatos = [NSMutableArray new];
[contatos addObject:contato];
```

E mandamos imprimir o *array*, não apenas um contato:

```
NSLog(@"%@", contatos);
```

Se rodarmos agora a simulação, ao adicionarmos, por exemplo, dois contatos, no log eles irão aparecer separados:

Mas, perceba que o contato adicionado anteriormente não aparece na segunda vez. Precisamos indicar para que não se crie um *array* novo, mas sim apenas um em que vamos adicionando objetos. Então declaramos o *array* no `initWithCoder`. Mas também queremos usá-lo dentro do método `adiciona`. Logo, em vez de utilizar uma variável local, instanciamos o *array* como uma propriedade:

```
-(id)initWithCoder: (NSCoder *) aDecoder {
    //...

        self.contatos = [NSMutableArray new];
    }
    return self;
}
```

E dentro de `adiciona`:

```
NSLog(@"%@", self.contatos);
```

No ViewController.h declaramos a propriedade do `NSMutableArray`:

```
@property NSMutableArray *contatos;
```

Fazendo o *build* e adicionando os mesmos contatos ainda não aparecem um seguido do outro no log. Acontece que o *array* está sendo único para a tela do formulário. Toda vez que apertamos o `+`, ou seja, o `exibeFormulario` ele está instanciando sempre um contato novo.

O formulário não pode ser o responsável pela criação desse *array*, mas sim apenas passar a informação. A responsabilidade tem que ser da lista. Logo, apagamos o `self.contatos...` do `initWithCoder` e o instanciamos na lista de contatos dentro do `init`:

```
-(id) init {
    //...
    if (self) {
        //...
        self.contatos = [NSMutableArray new];
    }
    return self;
}
```

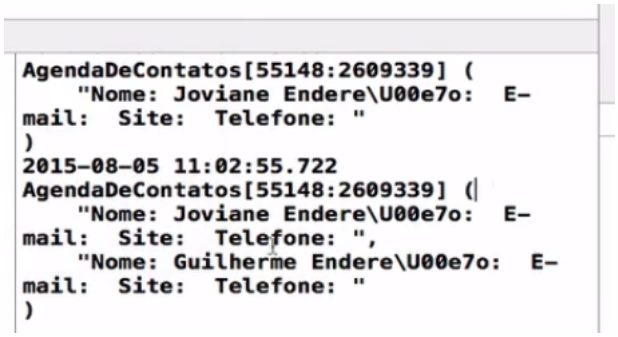
E criamos a propriedade no ListaContatosVoewController.h:

```
@property NSMutableArray *contatos;
```

Precisamos ainda dizer para a lista, em ListaContatosViewController.m:

```
-(void) exibeFormulario {
    //...
    form.contatos = self.contatos;
}
```

Rodando a simulação e, novamente, adicionando dois contatos:



Agora sim estão sendo guardados mais de um contato em nosso *array*

UITableViewCell