

App Agenda Persistência

Gravando e recuperando dados com SQLite sem uso do FrameWork
CoreData

X-Code com Objective-c
Prof. Agesandro Scarpioni

App - Agenda - SQLite

- Por default já vem instalado no Mac OS X um cliente de banco de dados, o cliente instalado é o SQLite.
- Para testarmos basta abrirmos o terminal e inserirmos os comandos para criação de banco, tabela, insert, select, etc.
- O SQLite possui diversos aplicativos para gerenciar o banco de forma visual como por exemplo o SQLite Data Browser.

App - Agenda - SQLite

- O local onde fica o banco quando o criamos no simulador do Iphone é:

```
///Users/< user >/Library/Application Support/Iphone Simulator/< versão do sdk >/Applications/< id da aplicação >/  
Documents/
```

- Veja o a imagem abaixo de um NSLog da URL do local de um banco qualquer:

```
2014-07-07 17:19:34.527 CoreData_ExemploUso[1813:60b] DB PATH file:///Users/  
agesandrosarpioni/Library/Application%20Support/iPhone%20Simulator/7.1/Applications/  
4E5BC306-68B1-4147-888D-017961E5A288/Documents/
```

Dica: Para acessar o pasta Library no Finder clique no menu IR com a tecla Option pressionada, a pasta Library irá aparecer.

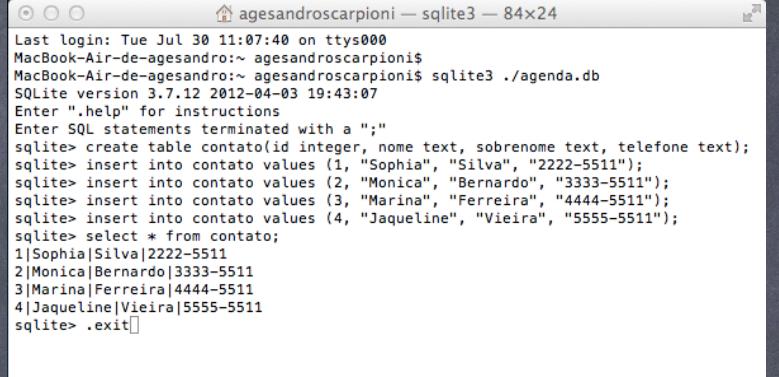
SQLite via Terminal



SQLite via Terminal

```
agesandroscarpioni — sqlite3 — 84x24
Last login: Tue Jul 30 11:07:40 on ttys000
MacBook-Air-de-agesandro:~ agesandroscarpioni$ MacBook-Air-de-agesandro:~ agesandroscarpioni$ sqlite3 ./agenda.db
SQLite version 3.7.12 2012-04-03 19:43:07
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> create table contato(id integer, nome text, sobrenome text, telefone text);
sqlite> insert into contato values (1, "Sophia", "Silva", "2222-5511");
sqlite> insert into contato values (2, "Monica", "Bernardo", "3333-5511");
sqlite> insert into contato values (3, "Marina", "Ferreira", "4444-5511");
sqlite> insert into contato values (4, "Jaqueline", "Vieira", "5555-5511");
sqlite> select * from contato;
1|Sophia|Silva|2222-5511
2|Monica|Bernardo|3333-5511
3|Marina|Ferreira|4444-5511
4|Jaqueline|Vieira|5555-5511
sqlite> .exit
```

SQLite via Terminal

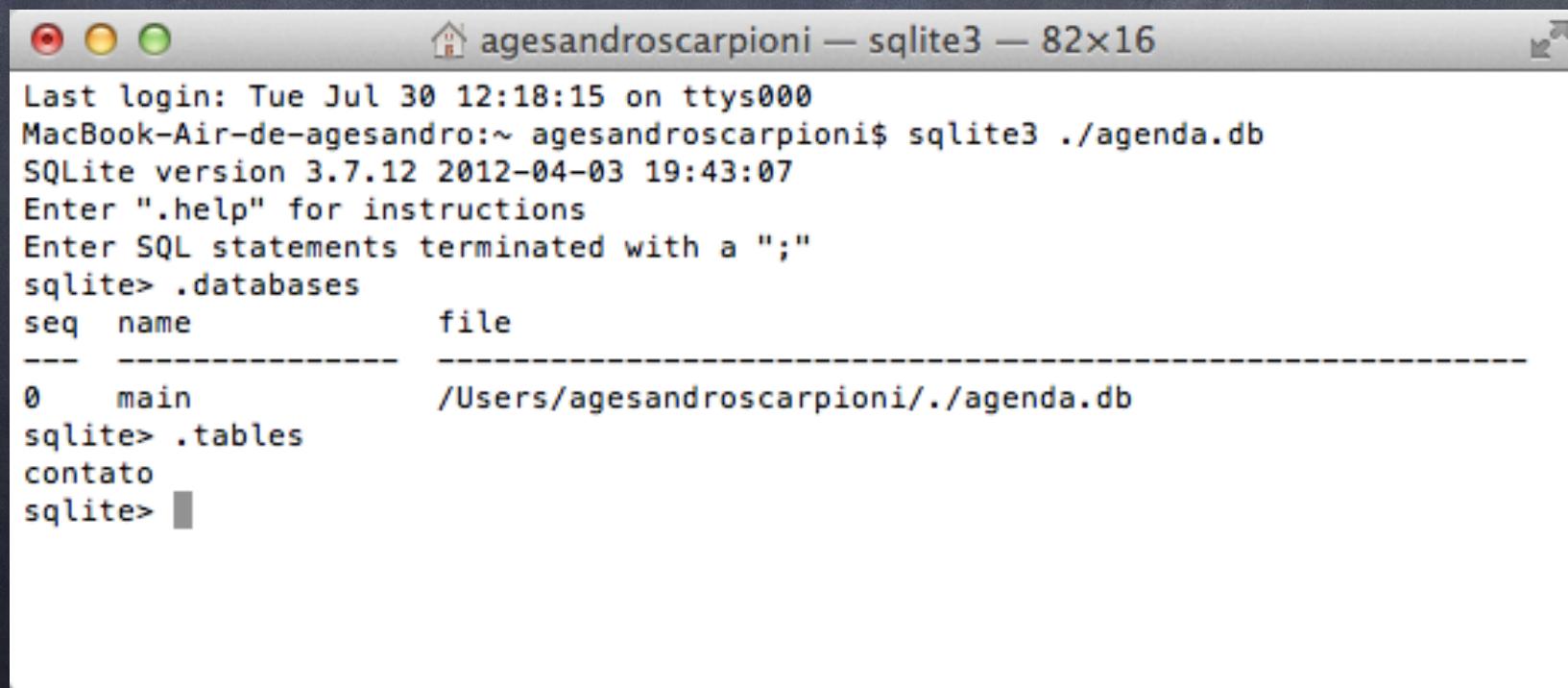


```
Last login: Tue Jul 30 11:07:40 on ttys000
MacBook-Air-de-agesandro:~ agesandrosarpioni$ 
MacBook-Air-de-agesandro:~ agesandrosarpioni$ sqlite3 ./agenda.db
SQLite version 3.7.12 2012-04-03 19:43:07
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> create table contato(id integer, nome text, sobrenome text, telefone text);
sqlite> insert into contato values (1, "Sophia", "Silva", "2222-5511");
sqlite> insert into contato values (2, "Monica", "Bernardo", "3333-5511");
sqlite> insert into contato values (3, "Marina", "Ferreira", "4444-5511");
sqlite> insert into contato values (4, "Jaqueline", "Vieira", "5555-5511");
sqlite> select * from contato;
1|Sophia|Silva|2222-5511
2|Monica|Bernardo|3333-5511
3|Marina|Ferreira|4444-5511
4|Jaqueline|Vieira|5555-5511
sqlite> .exit
```

- O comando “`sqlite3 ./agenda.db`” – indica onde e qual é o nome do banco que será criado.
- É padrão de outros bancos `create table`, `insert`, `select`, `delete`, `drop table <nome da tabela>`.
- `.exit` saímos do banco

SQLite via Terminal

- Entrando novamente com o comando: "sqlite3 ./agenda.db" podemos:
 - Ver o banco criado com o comando: .databases
 - Ver as tabelas do banco com: .tables

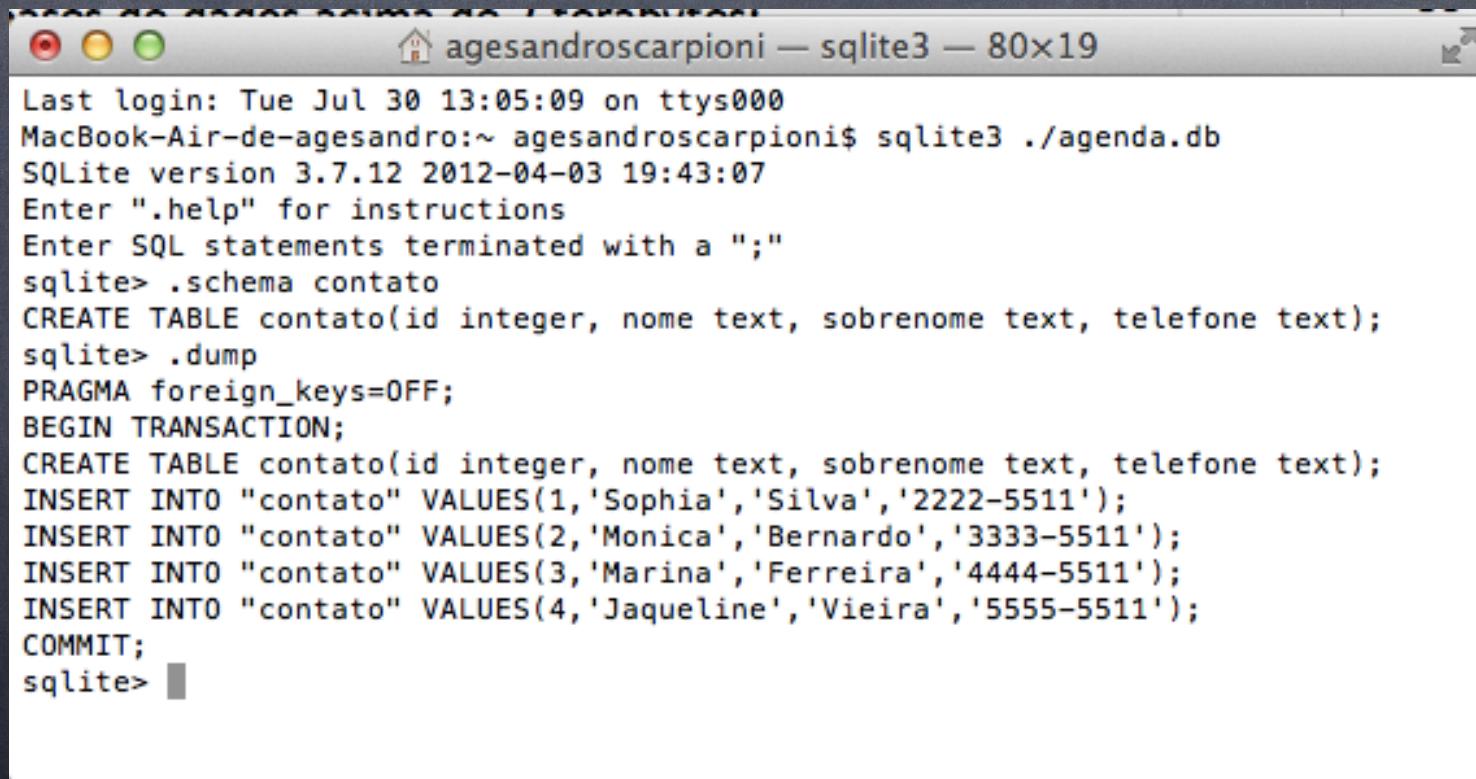


The screenshot shows a terminal window titled "agesandroscarpioni — sqlite3 — 82x16". The window contains the following text:

```
Last login: Tue Jul 30 12:18:15 on ttys000
MacBook-Air-de-agesandro:~ agesandroscarpioni$ sqlite3 ./agenda.db
SQLite version 3.7.12 2012-04-03 19:43:07
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> .databases
seq  name          file
---  -----
0    main          /Users/agesandroscarpioni./agenda.db
sqlite> .tables
contato
sqlite>
```

SQLite via Terminal

- .schema contato - mostra a estrutura da tabela contato
- .dump - mostra a estrutura e os dados da tabela.



```
Last login: Tue Jul 30 13:05:09 on ttys000
MacBook-Air-de-agesandro:~ agesandroscarpioni$ sqlite3 ./agenda.db
SQLite version 3.7.12 2012-04-03 19:43:07
Enter ".help" for instructions
Enter SQL statements terminated with a ";"

sqlite> .schema contato
CREATE TABLE contato(id integer, nome text, sobrenome text, telefone text);
sqlite> .dump
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE contato(id integer, nome text, sobrenome text, telefone text);
INSERT INTO "contato" VALUES(1,'Sophia','Silva','2222-5511');
INSERT INTO "contato" VALUES(2,'Monica','Bernardo','3333-5511');
INSERT INTO "contato" VALUES(3,'Marina','Ferreira','4444-5511');
INSERT INTO "contato" VALUES(4,'Jaqueline','Vieira','5555-5511');
COMMIT;
sqlite>
```

SQLite via Terminal

- Para apagar o banco não existe drop database, apague o arquivo diretamente da pasta onde foi ele foi criado.

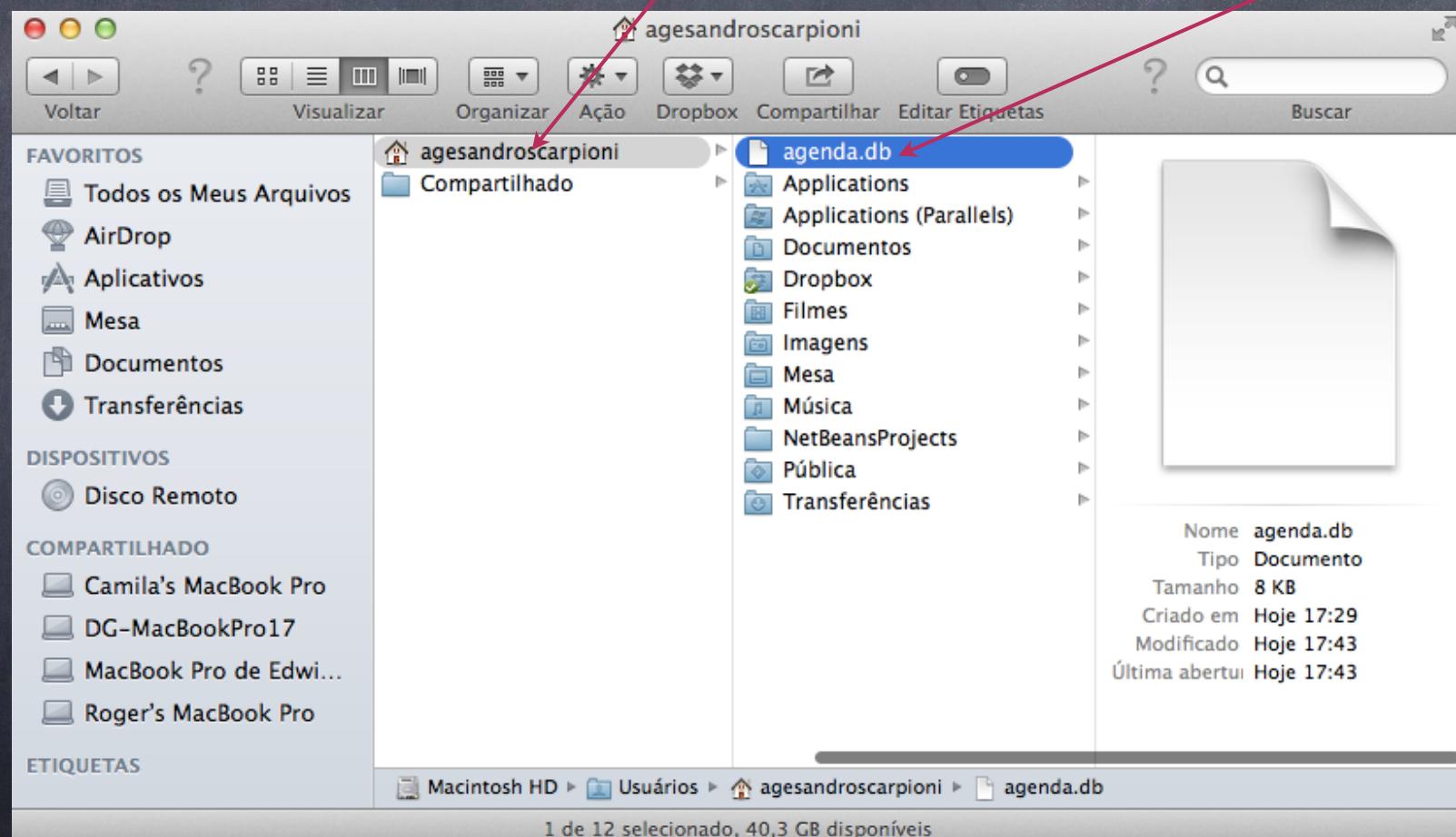


Table View

- Vamos criar um projeto novo do tipo IOS application (Single View Application) clique em Next.

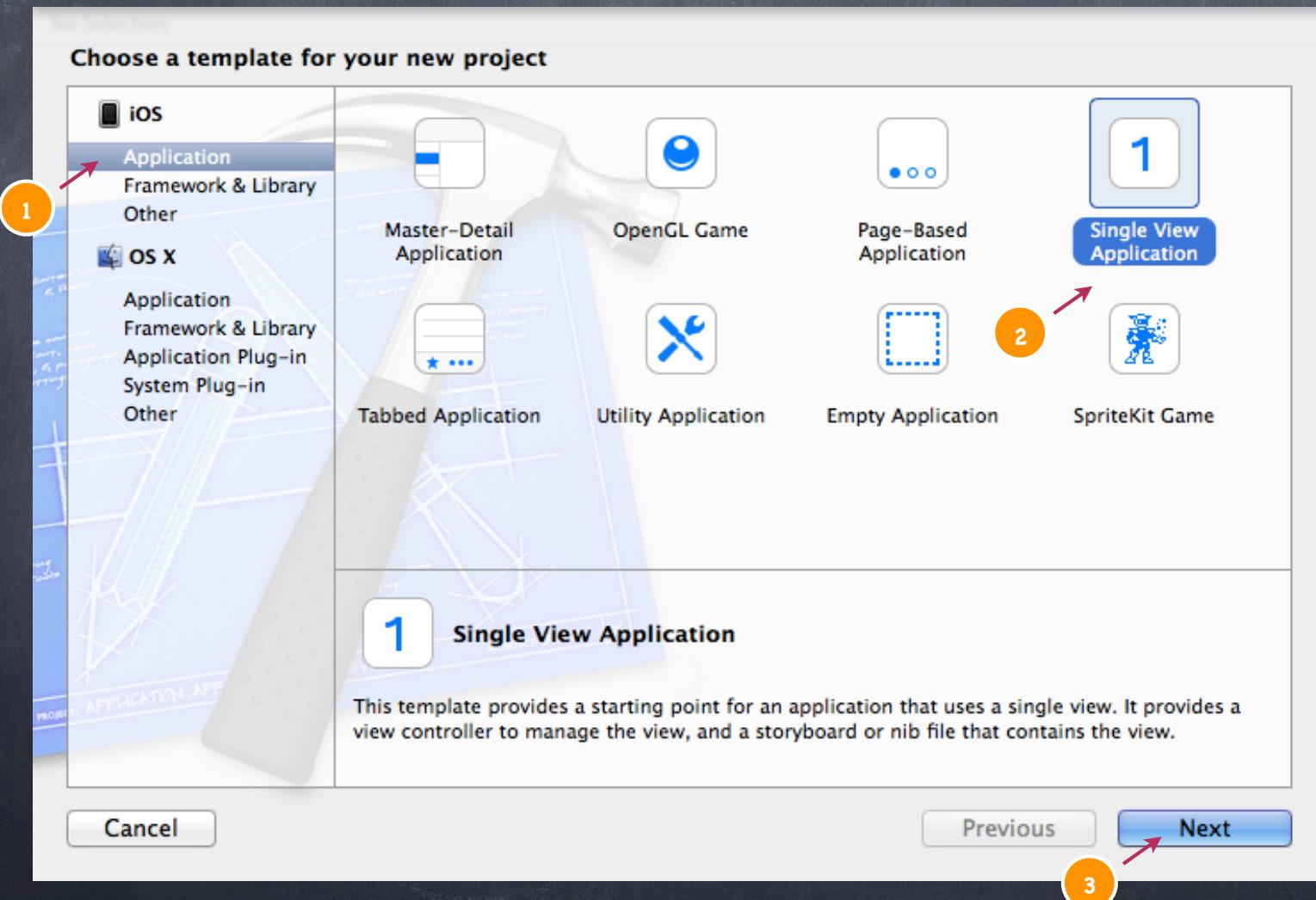
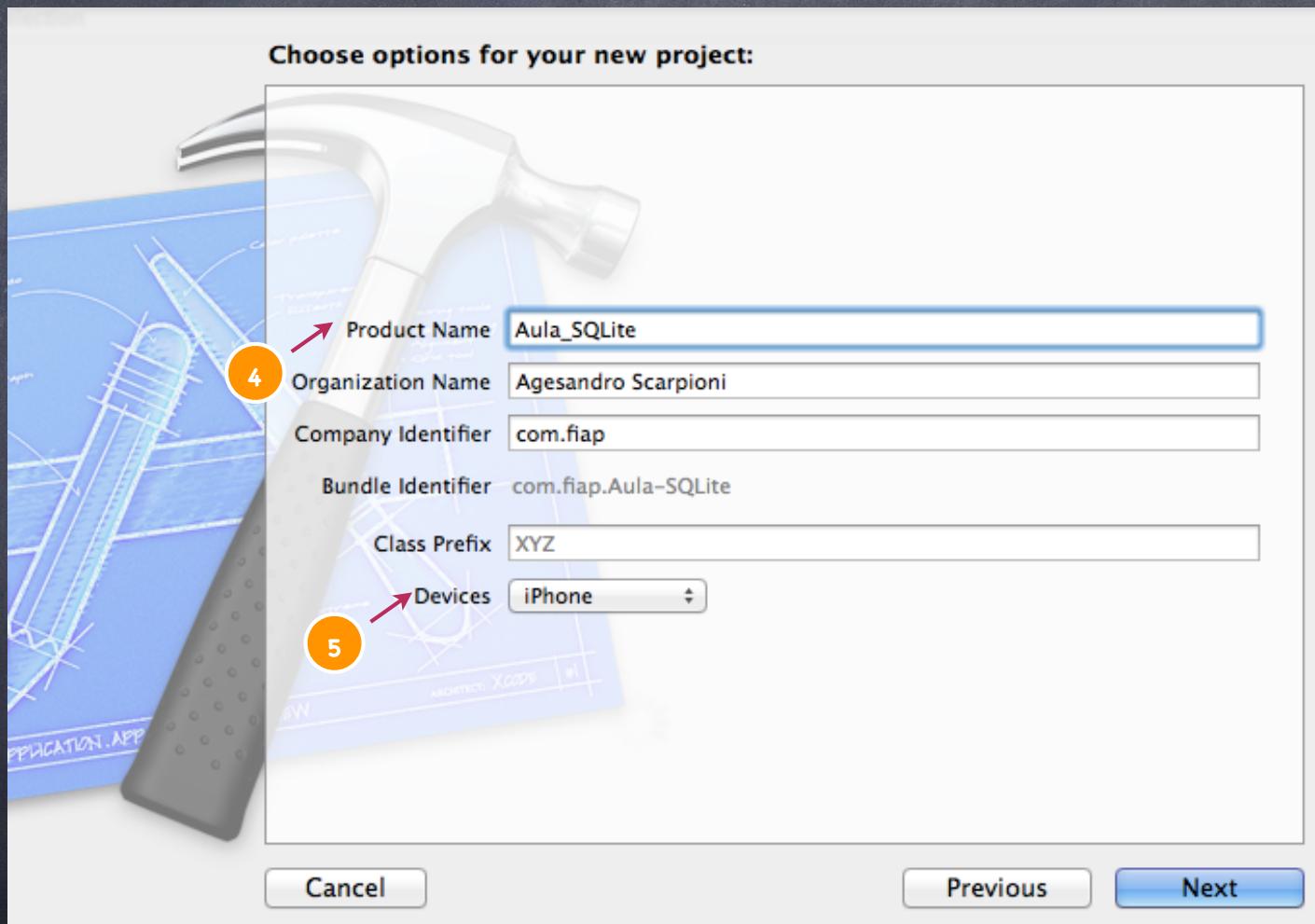


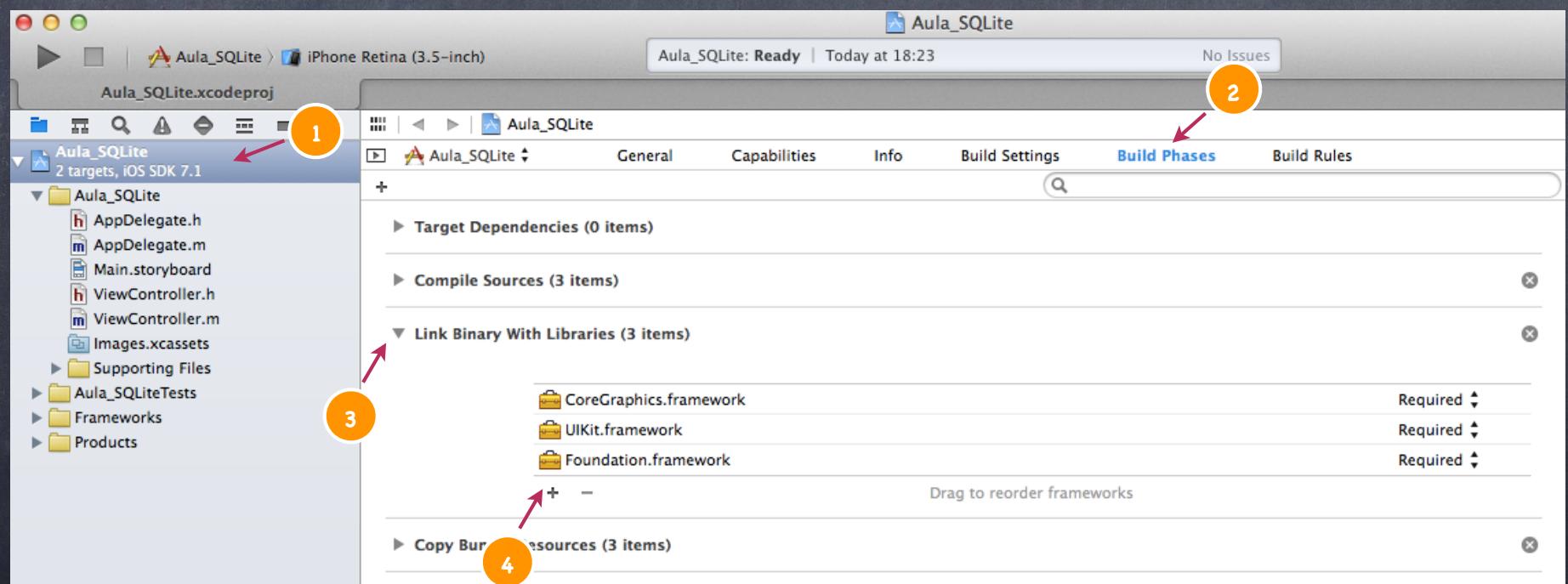
Table View

- Nomeie o projeto como: "Aula_SQLite" (4). Escolha o device iPhone(5).



Adicionar a biblioteca do SQLite

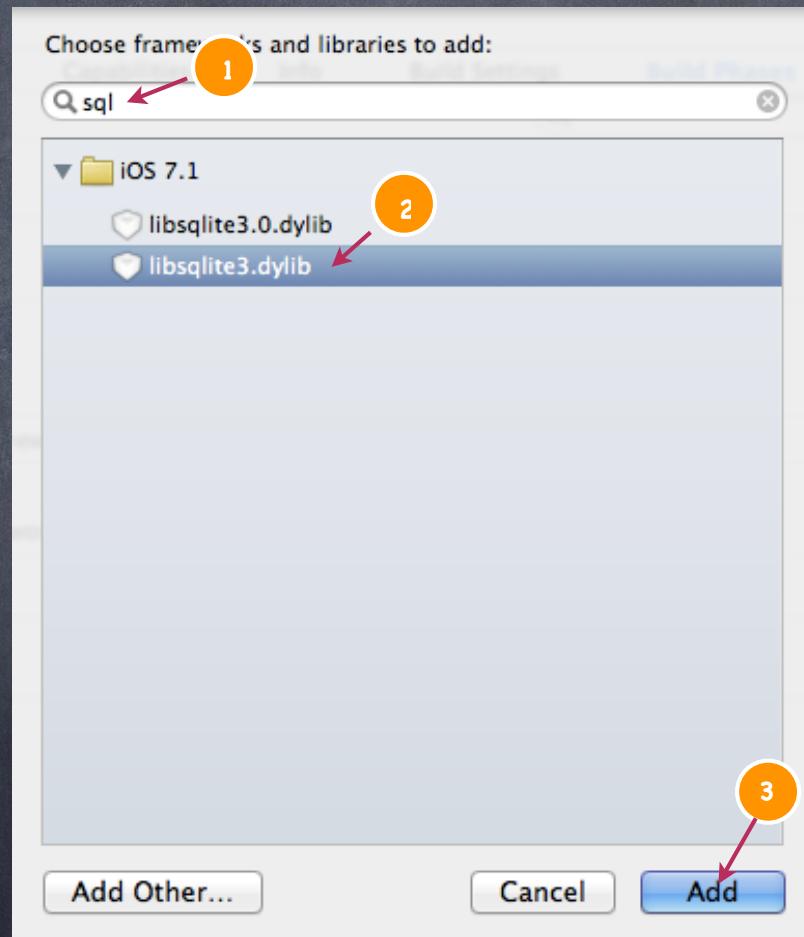
- Clique em Aula_SQLite target(1), clique em Build Phases(2), clique em (3) para abrir o Link clique no + (4) para selecionar a biblioteca.



OBS: Lembre-se de localizar o arquivo agenda.db no Mac e apagar o arquivo de banco de dados, vamos criar um novo banco com auto increment diretamente pela aplicação.

Adicionar a biblioteca do SQLite

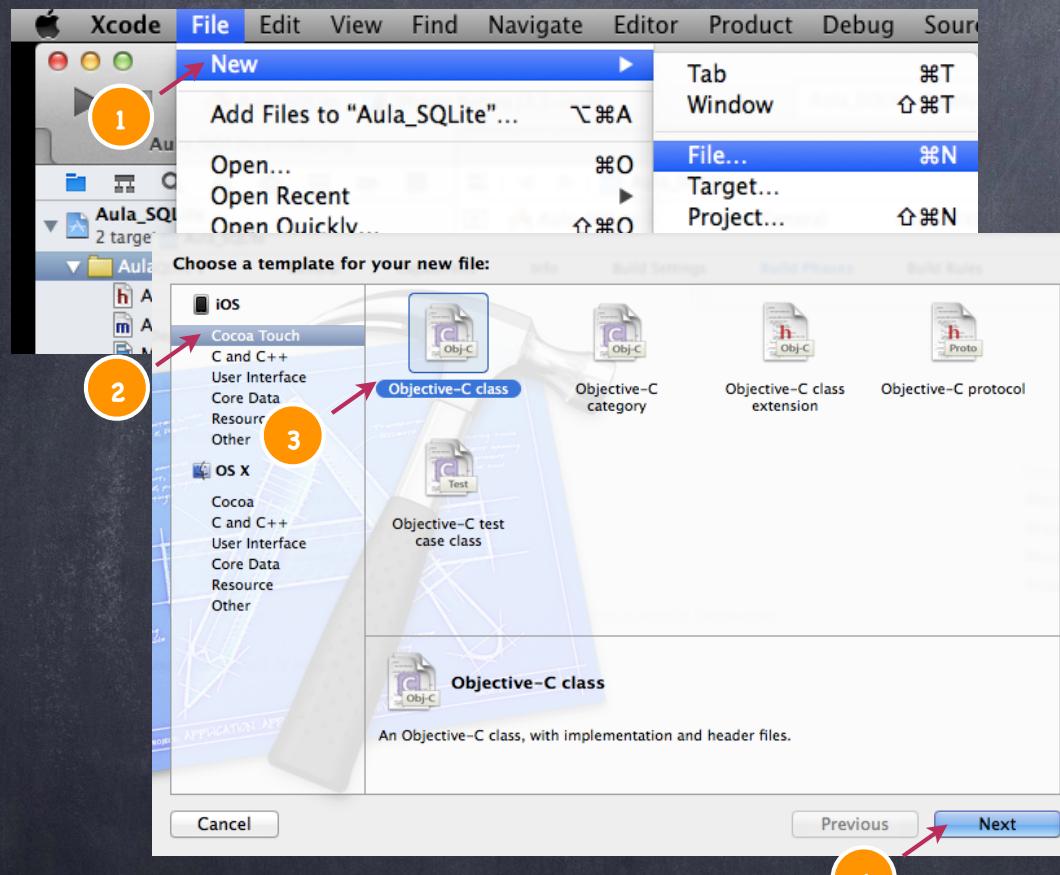
- Comece digitando sql... que irá aparecer libssqlite3.dylib



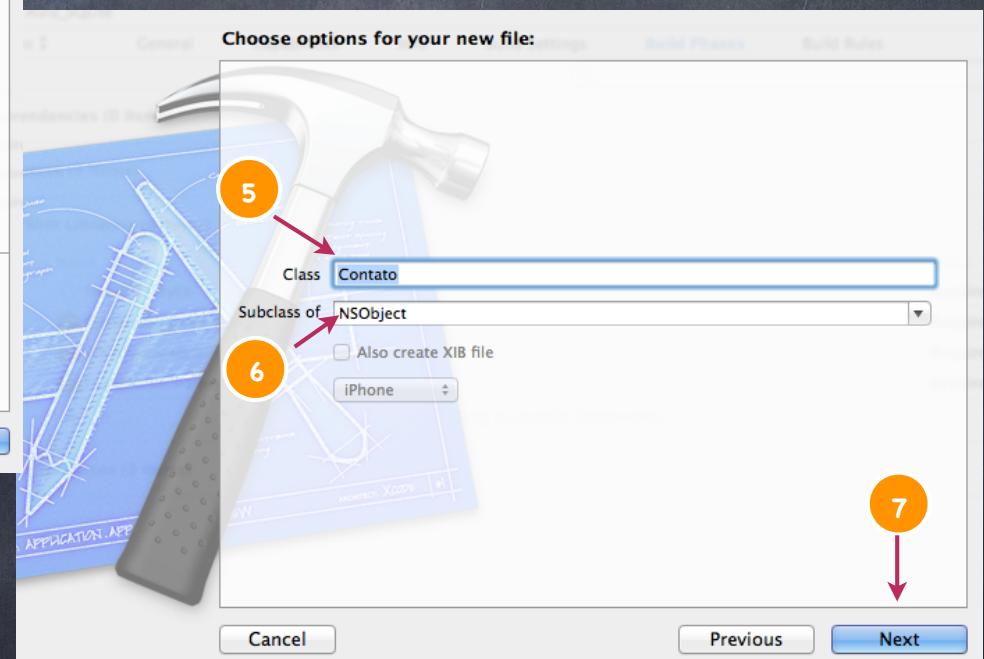
OBS: Para acessar os arquivos da biblioteca basta usar `#import "/usr/include/sqlite3.h"` ou apenas `#import "sqlite3.h"`

Classe Contato

- Vamos criar uma classe "Contato" para os métodos de get e set de nome, sobrenome e telefone. Clique em File->New->File-> IOS -> Objective-C class.



- Nomeie a classe como Contato
subclasse de NSObject



Classe Contato

- No Contato.h digite as chaves {} e inclua a programação abaixo para os get's e set's

```
1 //  
2 // Contato.h  
3 // Aula_SQLite  
4 //  
5 // Created by agesandro scarpioni on 07/07/14.  
6 // Copyright (c) 2014 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 #import <Foundation/Foundation.h>  
10  
11 @interface Contato : NSObject {  
12  
13 }  
14  
15 @property int codigo;  
16 @property (nonatomic ,retain) NSString *nome;  
17 @property (nonatomic ,retain) NSString *sobrenome;  
18 @property (nonatomic ,retain) NSString *telefone;  
19  
20 @end  
21
```

- Com a notação @property não precisamos declarar o set o get no arquivo.h nem sua implementação no arquivo .m, ela implementa automaticamente os getters e setters e ainda cuida do retain e do release, porém, para funcionar precisamos fazer o @synthesize das variáveis no arquivo .m, a partir da versão 4.4 do Xcode o @synthesize não é mais obrigatório

Mais sobre @property

20

```
@property (nonatomic, retain) NSString *nome;
```

- A palavra nonatomic indica que essa propriedade pode ser utilizada sem a necessidade de sincronizar as threads e portanto o acesso não é thread-safe. Como no iOS só existe uma thread por aplicação, é obrigatório utilizar o nonatomic, já no desenvolvimento para Mac OS X teríamos a opção atomic, que indica que o método é thread-safe e poderia ser utilizado de forma concorrente por diversas threads ao mesmo tempo.
- A palavra retain informa ao compilador para fazer o gerenciamento do retain e do release automaticamente, tirando essa responsabilidade do desenvolvedor.

Classe Contato

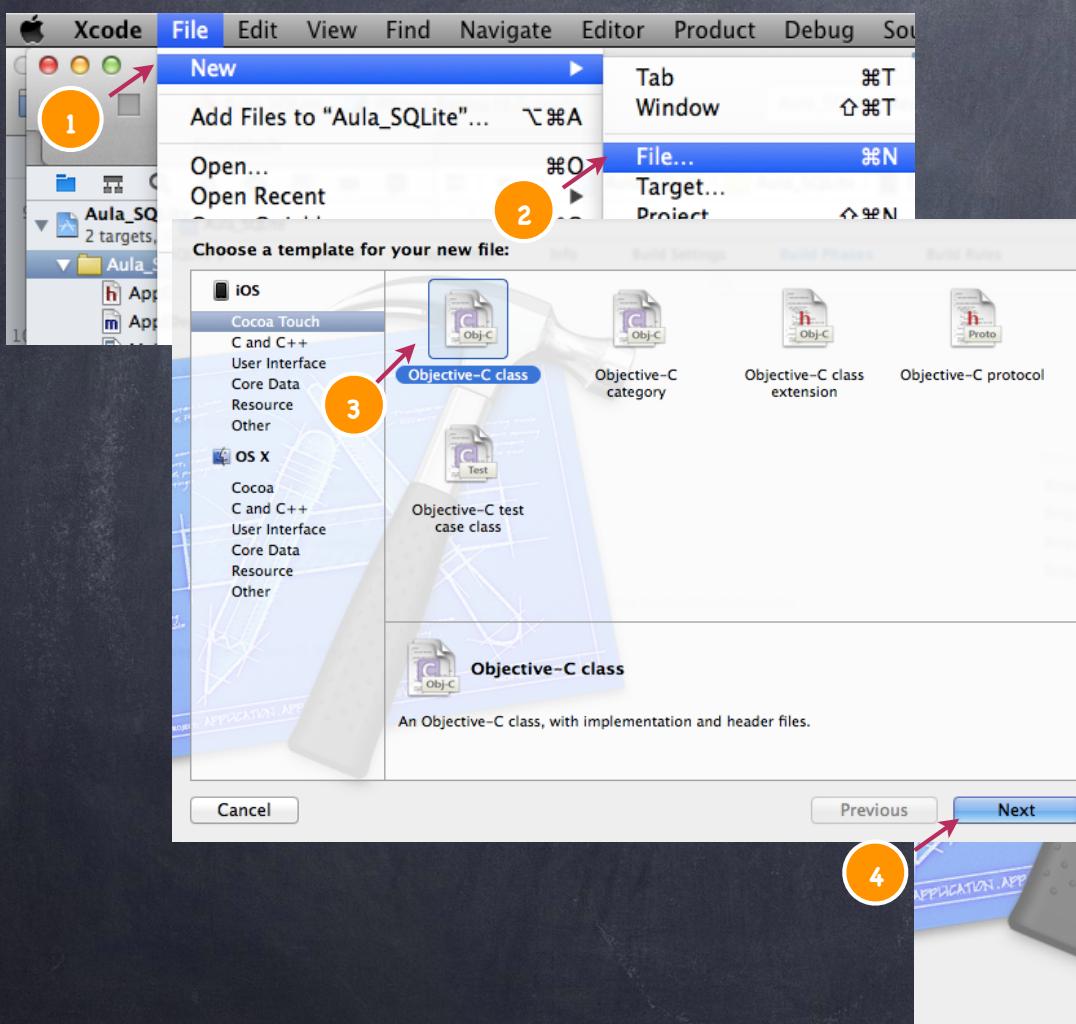
- No Contato.m digite a instrução @synthesize, ela indica ao compilador que deve implementar dinamicamente esses métodos para controlar o acesso ao objeto Contato.

```
1 //  
2 // Contato.m  
3 // Aula_SQLite  
4 //  
5 // Created by agesandro scarpioni on 07/07/14.  
6 // Copyright (c) 2014 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 #import "Contato.h"  
10  
11 @implementation Contato  
12  
13 @synthesize codigo, nome,sobrenome,telefone;  
14  
15 @end  
16
```

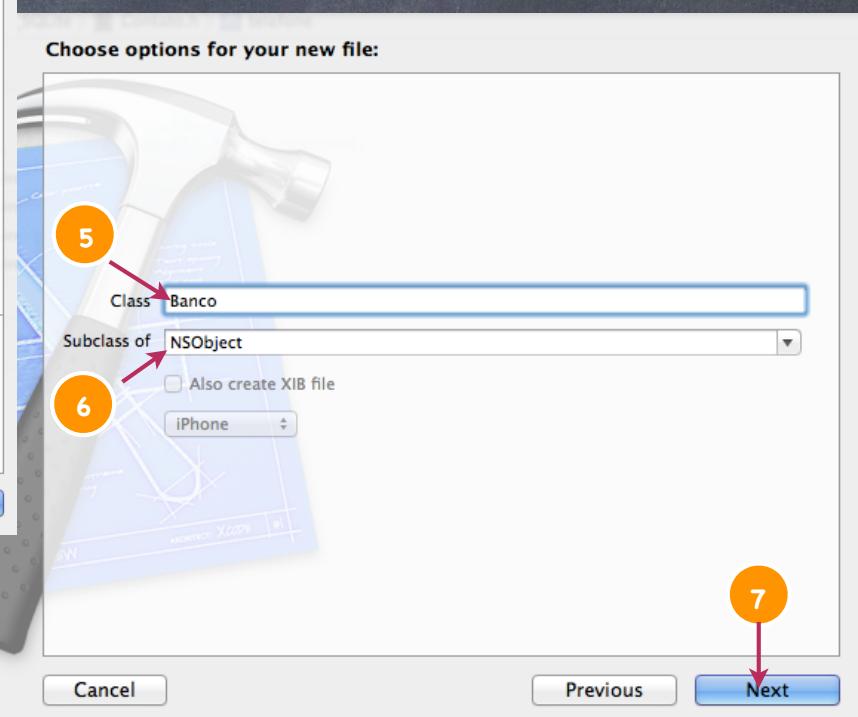
Dica: A vantagem de usar @property é que podemos utilizar a notação objeto.propriedade, além de internamente na classe usar self.propriedade diferenciando uma variável local de uma propriedade e facilitando a leitura.

Classe Banco

- Vamos criar uma classe “Banco” para abrir e fechar o banco, declarar o ponteiro “sqlite 3 *”, que é a referência para o bd aberto. Clique em File->New->File->IOS -> Objective-C class.



- Nomeie a classe como Banco
subclasse de NSObject



Classe Banco

- Na Banco.h vamos declarar os métodos de abrir, fechar banco e o retorno do SQL para criar a tabela.

```
1 //  
2 // Banco.h  
3 // Agenda1  
4 //  
5 // Created by agesandro scarpioni on 30/07/13.  
6 // Copyright (c) 2013 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 #import <Foundation/Foundation.h>  
10 #import "sqlite3.h"  
11  
12 @interface Banco : NSObject {  
13     sqlite3 *bancoDeDados; //Declaração do ponteiro para o banco de dados  
14 }  
15  
16 // abrir o banco de dados  
17 -(void) abrir:(NSString *) nomeBanco;  
18  
19 //fechar o banco  
20 -(void) fechar;  
21  
22 //Retorna o SQL para criar a tabela (deve ser implementado pela sub-classe)  
23 -(NSString *)getSQLParaCriarTabela;  
24  
25  
26 @end  
27
```

Classe Banco

- Na Banco.m vamos implementar os métodos de abrir, fechar banco e o método para retornar o caminho do BD.

```
1 //  
2 // Banco.m  
3 // Aula_SQLite  
4 //  
5 // Created by agesandro scarpioni on 07/07/14.  
6 // Copyright (c) 2014 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 #import "Banco.h"  
10  
11 @implementation Banco  
12 //este método devolve uma string montada com todo o caminho e nome do banco  
13 //o parâmetro de entrada é uma string com o nome do banco  
14 -(NSString *)caminhoDoArquivo:(NSString *)nomeBanco{  
15     //aqui contatenamos o nome do banco com a extensão sqlite3  
16     NSString *db = [NSString stringWithFormat:@"%@.sqlite3", nomeBanco];  
17     //cria o caminho do arq na pasta ..../documents  
18     NSArray *caminho = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);  
19     NSString *diretorioDocumento = [caminho objectAtIndex:0];  
20     //concatena o caminho com o nome do banco  
21     NSString *caminhoArquivo = [diretorioDocumento stringByAppendingPathComponent:db];  
22     NSLog(@"%@",caminhoArquivo);  
23     return caminhoArquivo;  
24 }  
25 }
```

Classe Banco

- Ainda no Banco.m vamos implementar apenas o método de abrir.

```
26 //abre o banco de dados no ponteiro "sqllite3 *bancoDeDados"
27 -(void) abrir:(NSString *) nomeBanco {
28
29     int result = sqlite3_open([[self caminhoDoArquivo:nomeBanco] UTF8String], &bancoDeDados);
30     if (result==SQLITE_OK){ // se o banco foi aberto com sucesso
31         //socilita a sub-classe para retornar o SQL de criação da tabela
32         NSString *sql = [self getSQLParaCriarTabela];
33         char *erroMsg;
34         //executa sql para criar a tabela, se necessário
35         int resultado = sqlite3_exec(bancoDeDados,[sql UTF8String],NULL,NULL,&erroMsg);
36         if (resultado == SQLITE_OK){
37             //tabela criada com sucesso
38         }else{
39             NSString *error = [NSString stringWithCString:erroMsg encoding:NSUTF8StringEncoding];
40             NSLog(@"Erro ao criar a tabela %d - %@", result, error);
41         }
42     }else {
43         NSLog(@"Erro ao abrir o banco de dados");
44     }
45 }
```

OBS: A função sqlite3_open cria ou abre um banco se já existir, ela recebe como parâmetro uma const char* que é uma string da linguagem C, por isso para passar uma NSString como parâmetro, primeiro precisamos chamar o método [NSString UTF8String] para converter o NSString do caminho+banco para char da linguagem C.

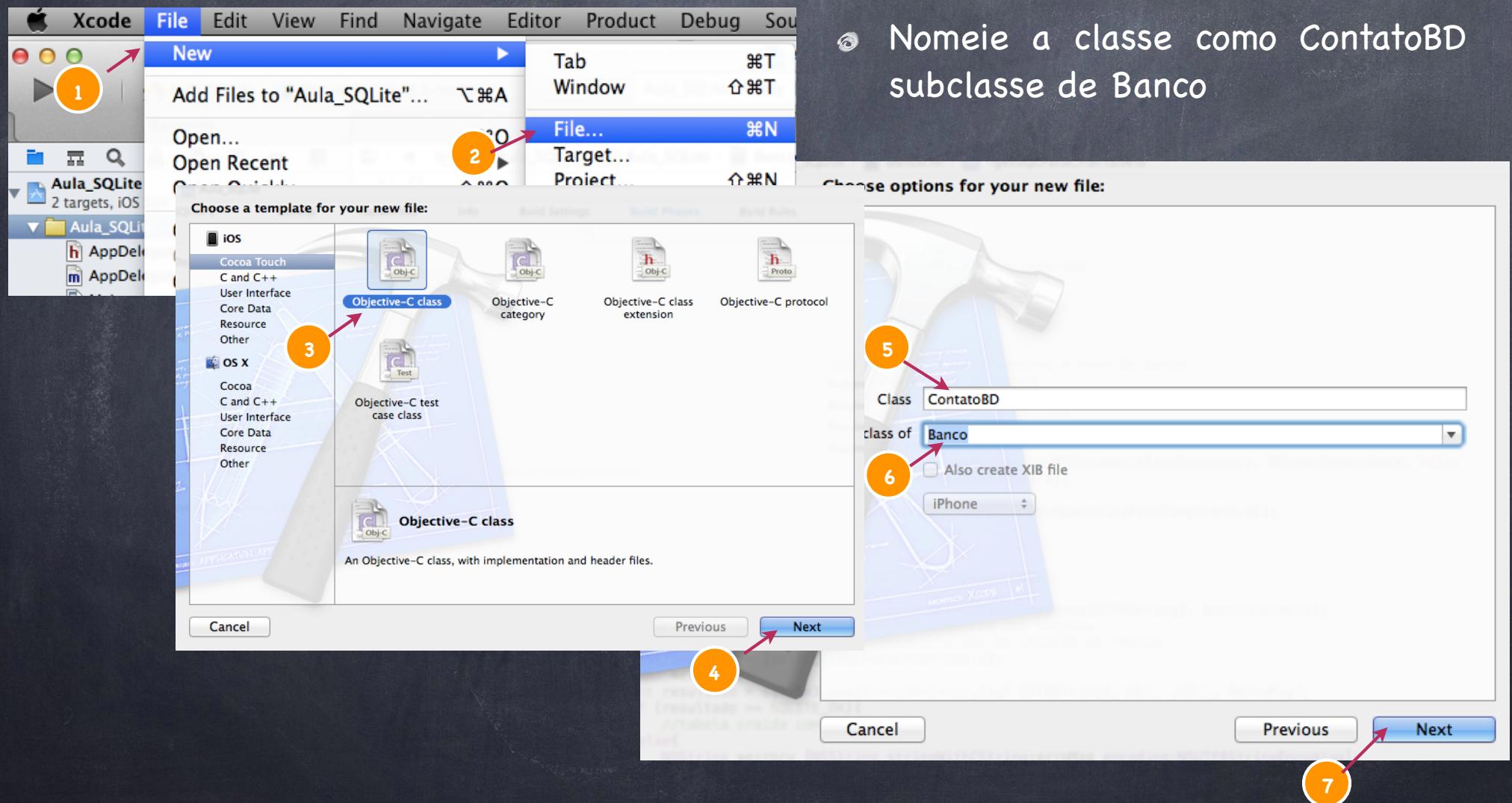
Classe Banco

- Ainda no Banco.m vamos implementar os métodos de fechar e retorno do sql para criar a tabela, para facilitar copie a declaração no Banco.h cole aqui antes do @end, retire ; e coloque { }.

```
46
47 //fechar o banco
48 -(void) fechar {
49     sqlite3_close(bancoDeDados);
50 }
51
52 //Retorna o SQL para criar a tabela
53 -(NSString *)getSQLParaCriarTabela{
54     // as sub-classes precisam implementar;
55     return nil;
56 }
57
58 @end
59
```

Classe ContatoBD

- Vamos criar uma classe "ContatoBD" sub-classe de "Banco" para incluir, alterar, excluir contatos. Clique em File->New->File-> IOS -> Objective-C class.



Classe ContatoBD

- Na ContatoBD.h vamos declarar os métodos de salvar e o método que retorna o SQL para criar a tabela(2).

```
1 //  
2 // ContatoBD.h  
3 // Aula_SQLite  
4 //  
5 // Created by agesandro scarpioni on 11/07/14.  
6 // Copyright (c) 2014 Agesandro Scarpioni. All rights reserved.  
7 //  
8 #import <Foundation/Foundation.h>  
9 #import "Banco.h"  
10 #import "Contato.h"  
11  
12  
13 @interface ContatoBD : Banco{  
14 }  
15 //Retorna o SQL para crair a tabela (deve ser implementado pela sub-classe  
16 -(NSString *)getSQLParaCriarTabela;  
17  
18 //Salva um novo contato ou atualiza um já existente  
19 -(void) salvar:(Contato *)contato;  
20  
21  
22 @end  
23
```

OBS: Como criamos a classe ContatoBD filha de Banco, faça o import do Foundation, além do contato (1).

Classe ContatoBD

- No ContatoBD.m traga a declaração getSQLParaCriarTabela, cole aqui, retire o ; e coloque as { }, em seguida faça a implementação abaixo:

```
1 //  
2 // ContatoBD.m  
3 // Aula_SQLite  
4 //  
5 // Created by agesandro scarpioni on 11/07/14.  
6 // Copyright (c) 2014 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 #import "ContatoBD.h"  
10  
11 @implementation ContatoBD  
12  
13 //Retorna o SQL para criar a tabela  
14 -(NSString *)getSQLParaCriarTabela{  
15     NSString *sql = @"create table if not exists contato(codigo integer primary key autoincrement, nome text, sobrenome text, telefone text);";  
16     return sql;  
17 }  
18
```

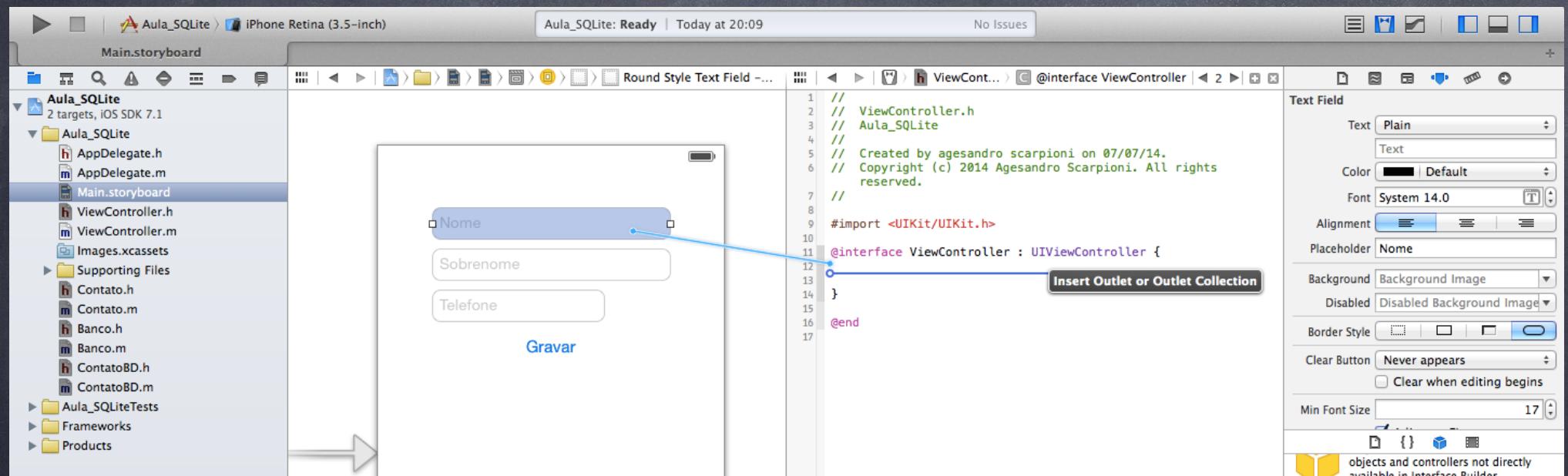
Classe ContatoBD

- Ainda no ContatoBD.m, traga a declaração do salvar da classe contatoBD.h, cole aqui, retire o ; e coloque as { }, em seguida faça a implementação abaixo:

```
19 //Salva um novo contato ou atualiza um já existente pelo codigo
20 -(void) salvar:(Contato *) contato{
21     char *sql = "insert or replace into contato (codigo, nome, sobrenome, telefone) values (?,?,?,?,?);";
22     sqlite3_stmt *stmt;
23     //a linha abaixo cria um statement para ser executado, -1 é a qtd de caracteres para ser
24     //lido, se valor -1 o sql é lido inteiro o &stmt é o statement que é o parâmetro de saída desta função.
25     int resultado = sqlite3_prepare_v2(bancoDeDados, sql,-1,&stmt,nil);
26     // a função sqlite3_prepare_v2 é igual a sqlite3_exec, porém com a sqlite3_prepare_v2
27     //precisamos do sqlite3_step e do sqlite3_finalize
28     if (resultado==SQLITE_OK){
29         //se for nulo insere, senão atualiza
30         if (contato.codigo >0){
31             //informa o codigo para fazer update
32             sqlite3_bind_int(stmt,1,contato.codigo);
33         }
34         sqlite3_bind_text(stmt,2, [contato.nome UTF8String],-1,nil);
35         sqlite3_bind_text(stmt,3, [contato.sobrenome UTF8String],-1,nil);
36         sqlite3_bind_text(stmt,4, [contato.telefone UTF8String],-1,nil);
37
38         // Executa o sql
39         resultado = sqlite3_step(stmt);
40         if (resultado == SQLITE_DONE){
41             // Inserido com sucesso
42             NSLog(@"INSERIDO COM SUCESSO");
43         }
44         sqlite3_finalize(stmt);
45     }else{
46         NSLog(@"Erro ao inserir contato %d", resultado);
47         return;
48     }
49 }
50
51 }
```

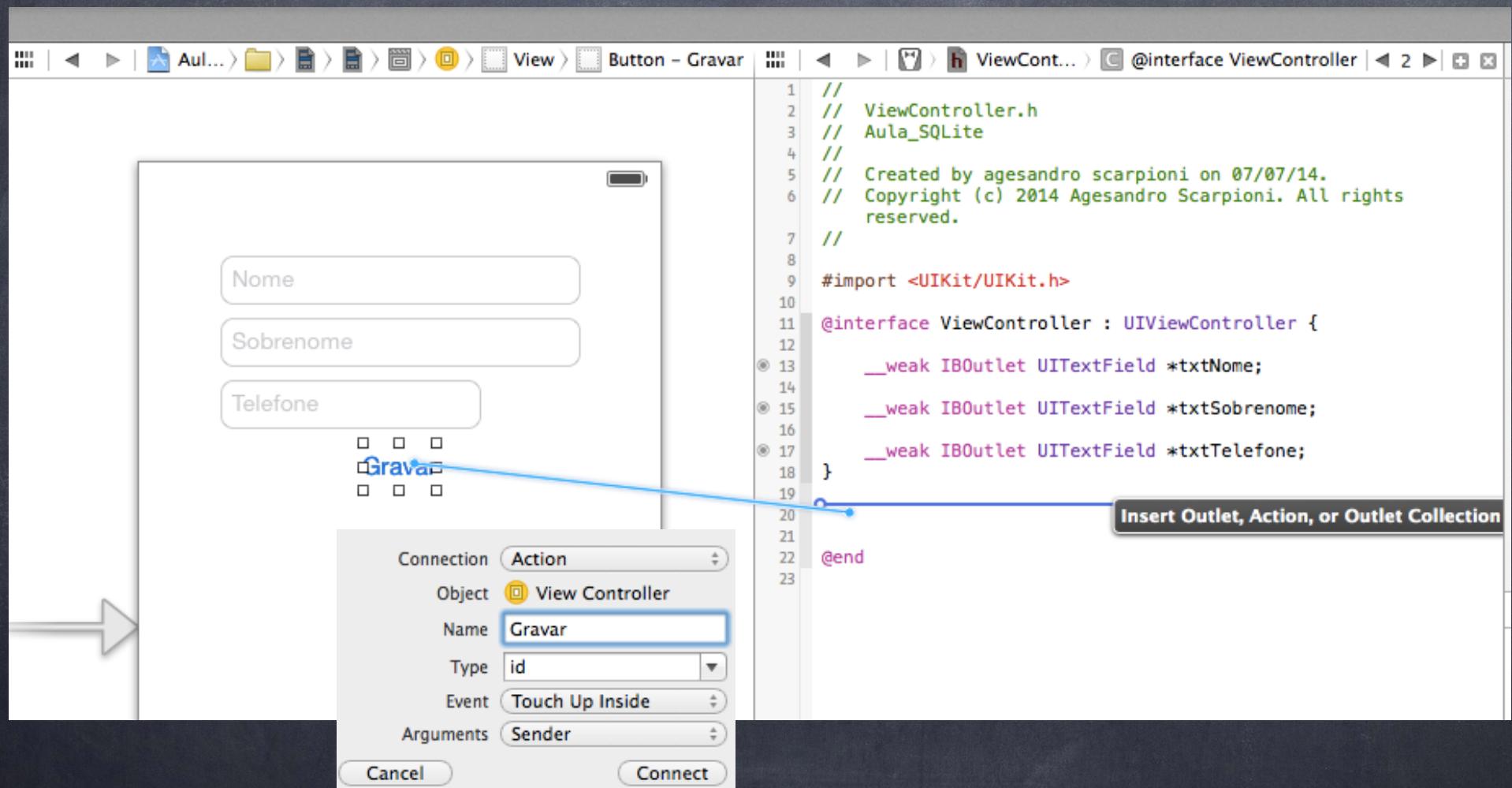
Storyboard

- Crie no storyboard 3 caixas de texto e 1 botão, posicione duas telas abra as chaves, crie 3 Outlet's e fora das chaves crie um Action para o Gravar. Veja o próximo slides os nomes de cada Outlet e Action.



Storyboard

- Veja os 3 Outlet's criados e observe o nome criado para o Action fora das {}.



ViewController.h

- Veja como ficou os Outlets e o Action

```
1 //  
2 // ViewController.h  
3 // Aula_SQLite  
4 //  
5 // Created by agesandro scarpioni on 11/07/14.  
6 // Copyright (c) 2014 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 #import <UIKit/UIKit.h>  
10  
11 @interface ViewController : UIViewController{  
12  
⑩ 13     __weak IBOutlet UITextField *txtNome;  
14  
⑩ 15     __weak IBOutlet UITextField *txtSobrenome;  
16  
⑩ 17     __weak IBOutlet UITextField *txtTelefone;  
18  
19 }  
20  
⑩ 21 - (IBAction)gravar:(id)sender;  
22  
23 @end  
24
```

ViewController.m

- Faça o import de ContatoBD.h (1) implemente as linhas abaixo no Gravar(1).

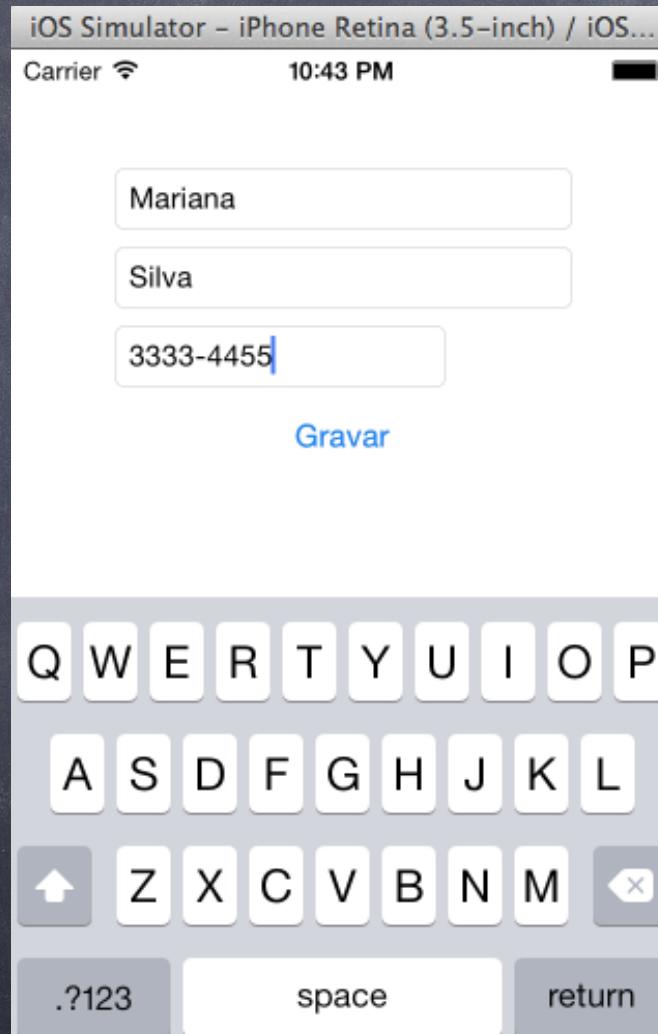
```
1 //  
2 //  ViewController.m  
3 //  Aula_SQLite  
4 //  
5 //  Created by agesandro scarpioni on 11/07/14.  
6 //  Copyright (c) 2014 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 #import "ViewController.h"  
10 #import "ContatoBD.h"  
11  
12 @interface ViewController ()  
13  
14 @end  
15  
16 @implementation ViewController  
17  
18 - (void)viewDidLoad  
19 {  
20     [super viewDidLoad];  
21     // Do any additional setup after loading the view, typically from a nib.  
22 }  
23  
24 - (void)didReceiveMemoryWarning  
25 {  
26     [super didReceiveMemoryWarning];  
27     // Dispose of any resources that can be recreated.  
28 }  
29  
30 - (IBAction)gravar:(id)sender {  
31     ContatoBD *db = [[ContatoBD alloc] init];  
32     Contato *contato = [[Contato alloc] init];  
33     [db abrir:@"agenda"]; //abre o banco  
34     contato.nome = txtNome.text;  
35     contato.sobrenome = txtSobrenome.text;  
36     contato.telefone = txtTelefone.text;  
37     [db salvar:contato];  
38     [db fechar];  
39 }  
40  
41 @end
```

1

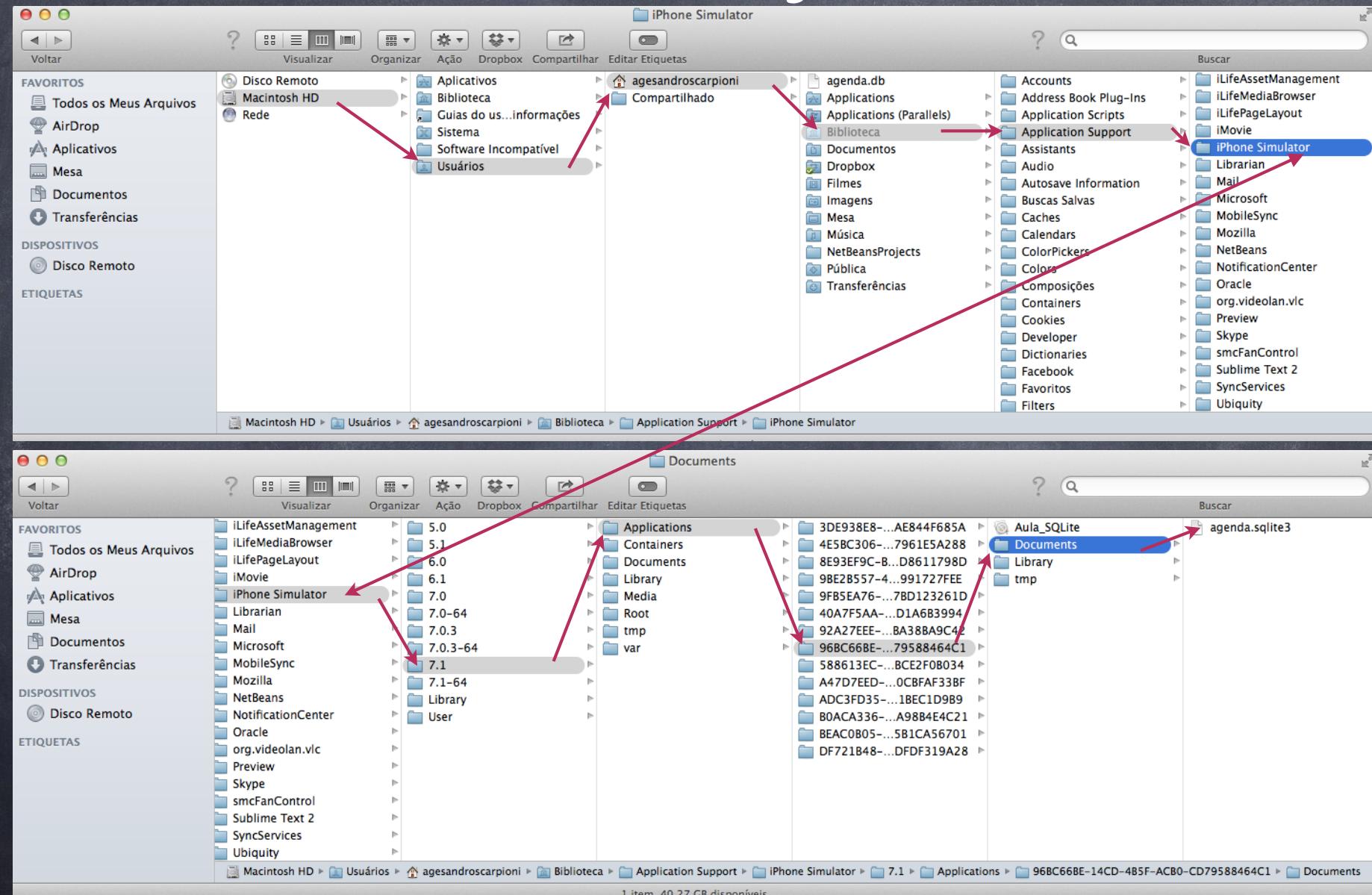
Testando

- Execute o aplicativo Command + R, digite um nome, um sobrenome e um telefone. Feche o aplicativo.
- Pronto os dados foram salvos, faça o teste com alguns nomes.
- Veja no próximo slide o banco (gerado em tempo de execução) na pasta informada nos primeiros slides.

Testando

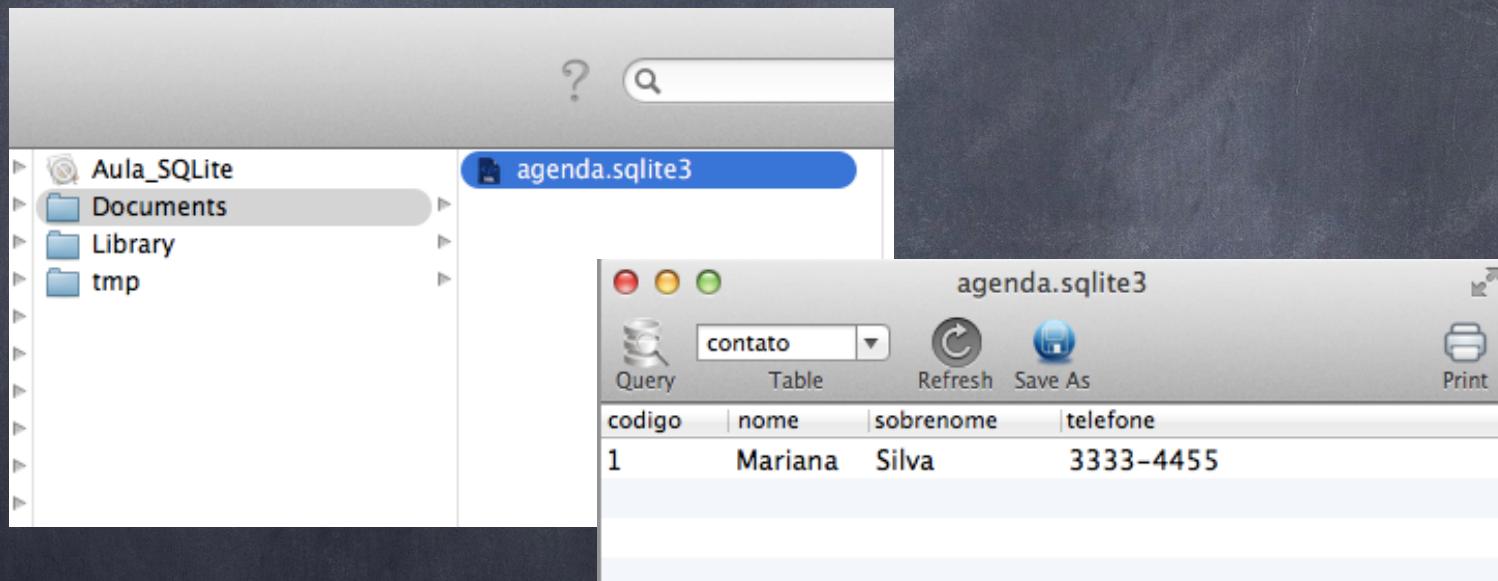


O banco criado em tempo de execução



agenda.sqlite3

- Clicando 2x sobre o banco e utilizando o App Database baixado na Apple Store você pode visualizar o conteúdo do banco.



App Agenda Persistência

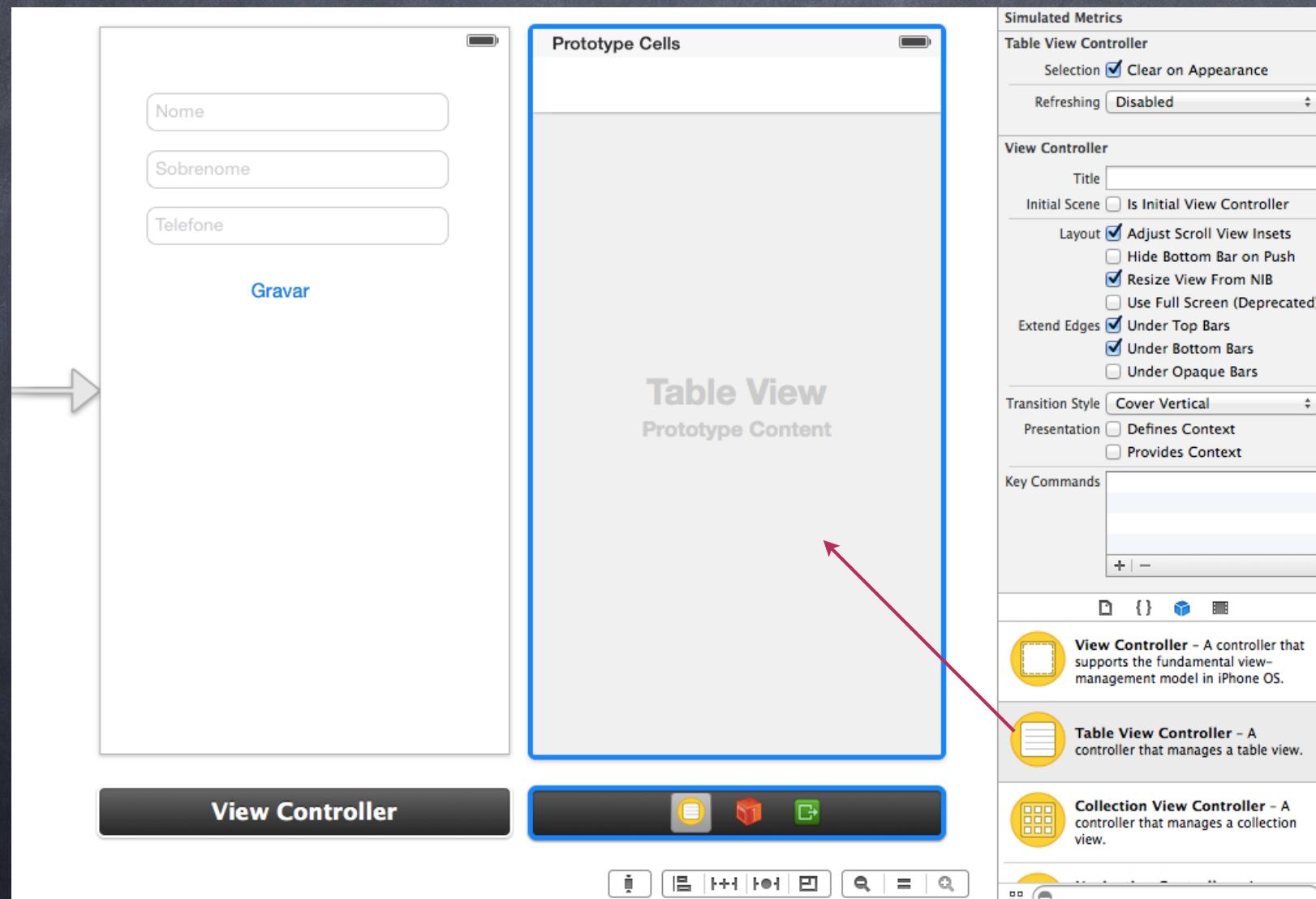
Parte 2 - Exibir os dados cadastrados em um TableView com SQLite

X-Code

Prof. Agesandro Scarpioni

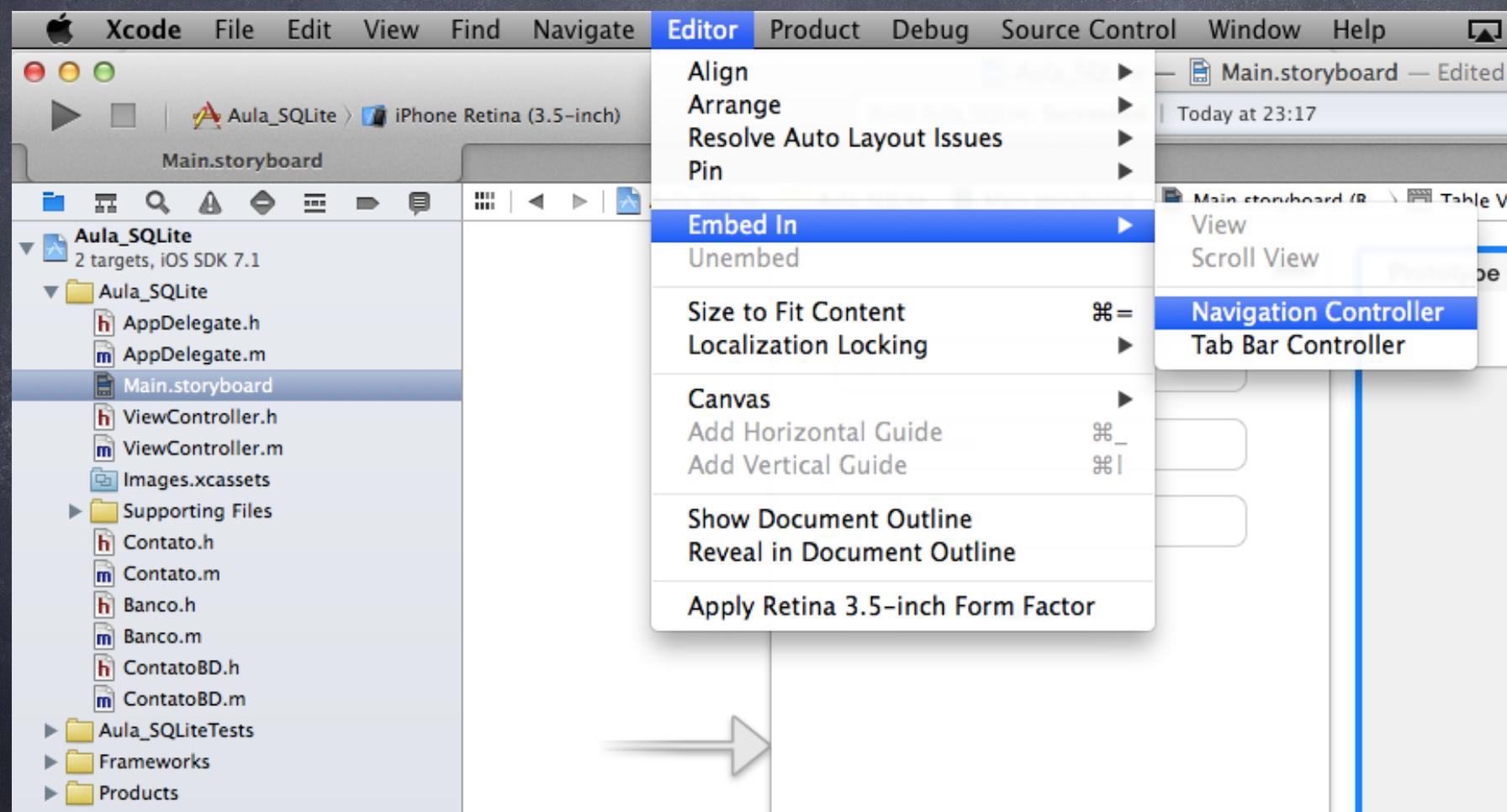
Exibir os Dados

- Vamos preparar a interface para exibir os dados, inclua no Storyboard um Table View Controller



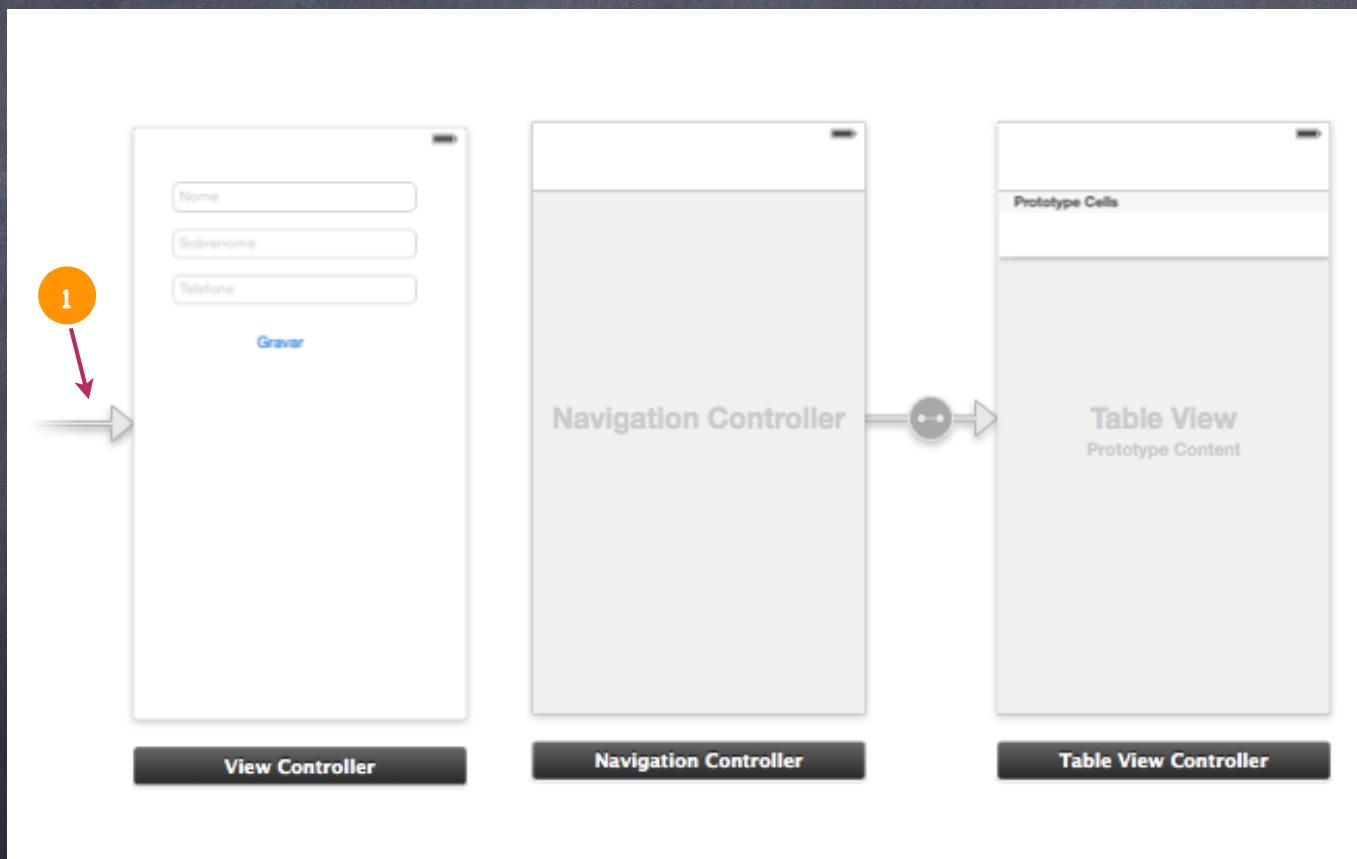
Exibir os Dados

- Vamos incluir um Navigation Controller para gerenciar as duas telas, deixe o tableViewController selecionado e clique em: Editor → Embed In → Navigation Controller.



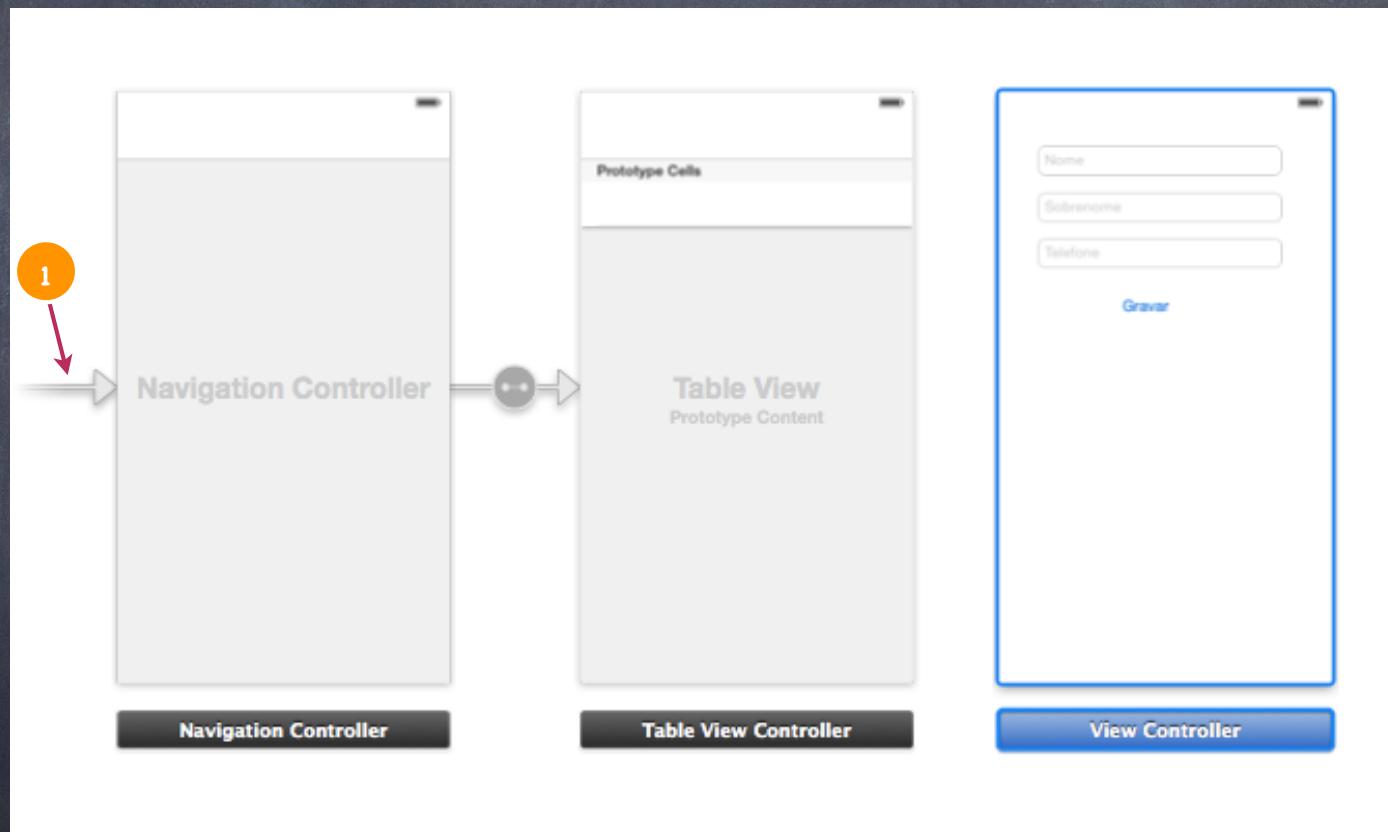
Exibir os Dados

- Observe que após essa etapa o Navigation Controller está ligado ao Table View, porém, a seta que indica qual tela irá aparecer quando executarmos o APP está na ViewController (1), precisamos arrastar essa seta para o Navigation Controller, como poderá ser visto no próximo slide.



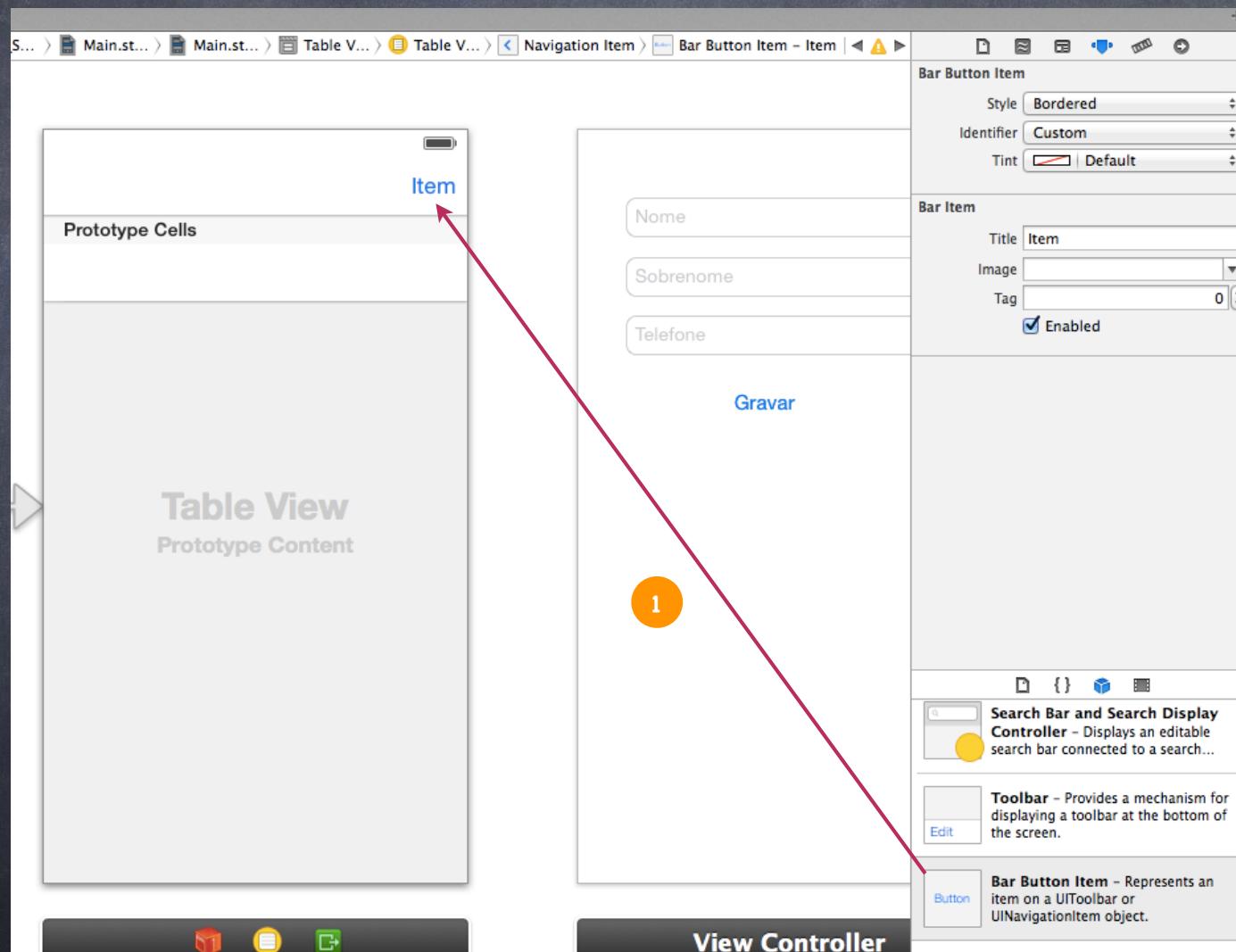
Exibir os Dados

- Além de passar a seta para o Navigation Controller passamos a tela de gravar para ficar após o Table View Controller.



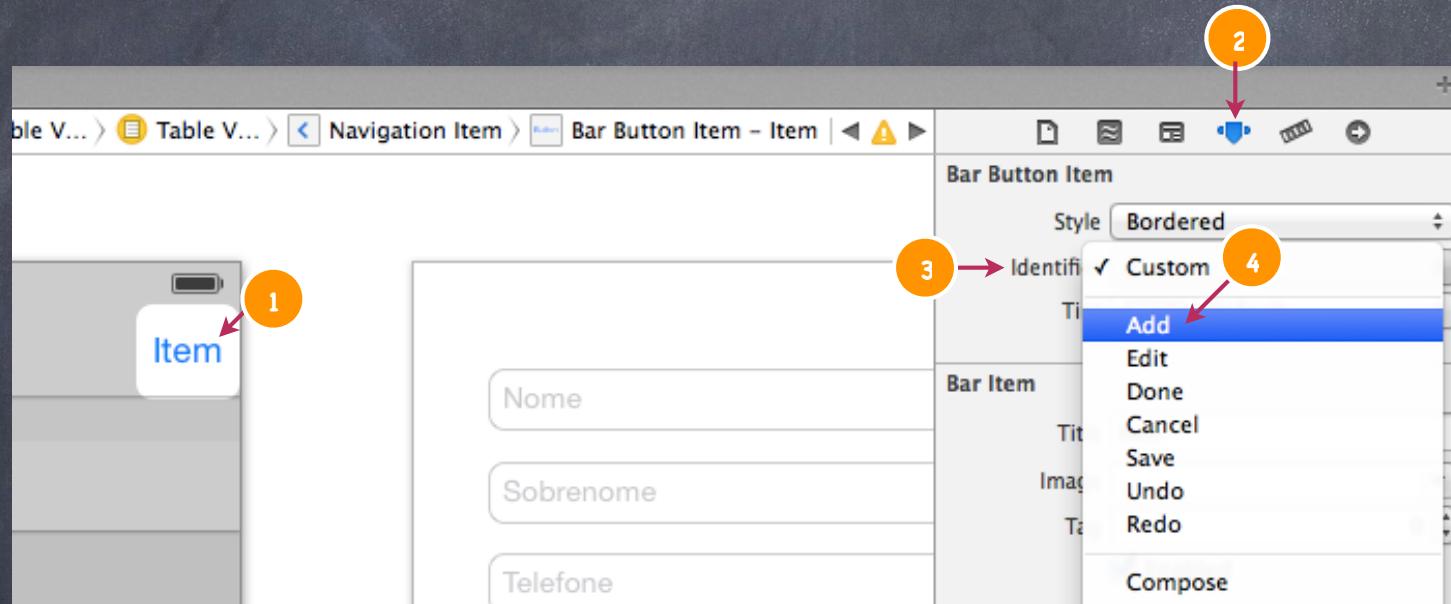
Exibir os Dados

- Insira um Bar Button Item na Barra de Navegação do Table View Controller (1).



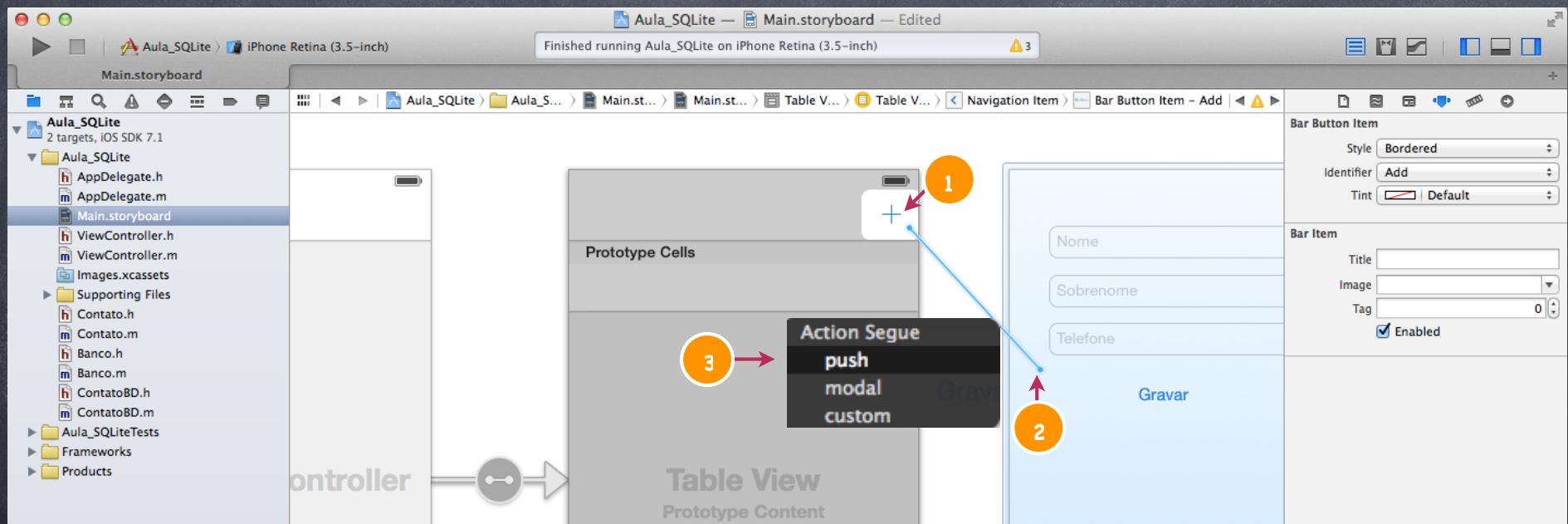
Exibir os Dados

- Selecione o Bar Button Item(1) e nas propriedades (2) altere o Identifier (3) para Add (4).



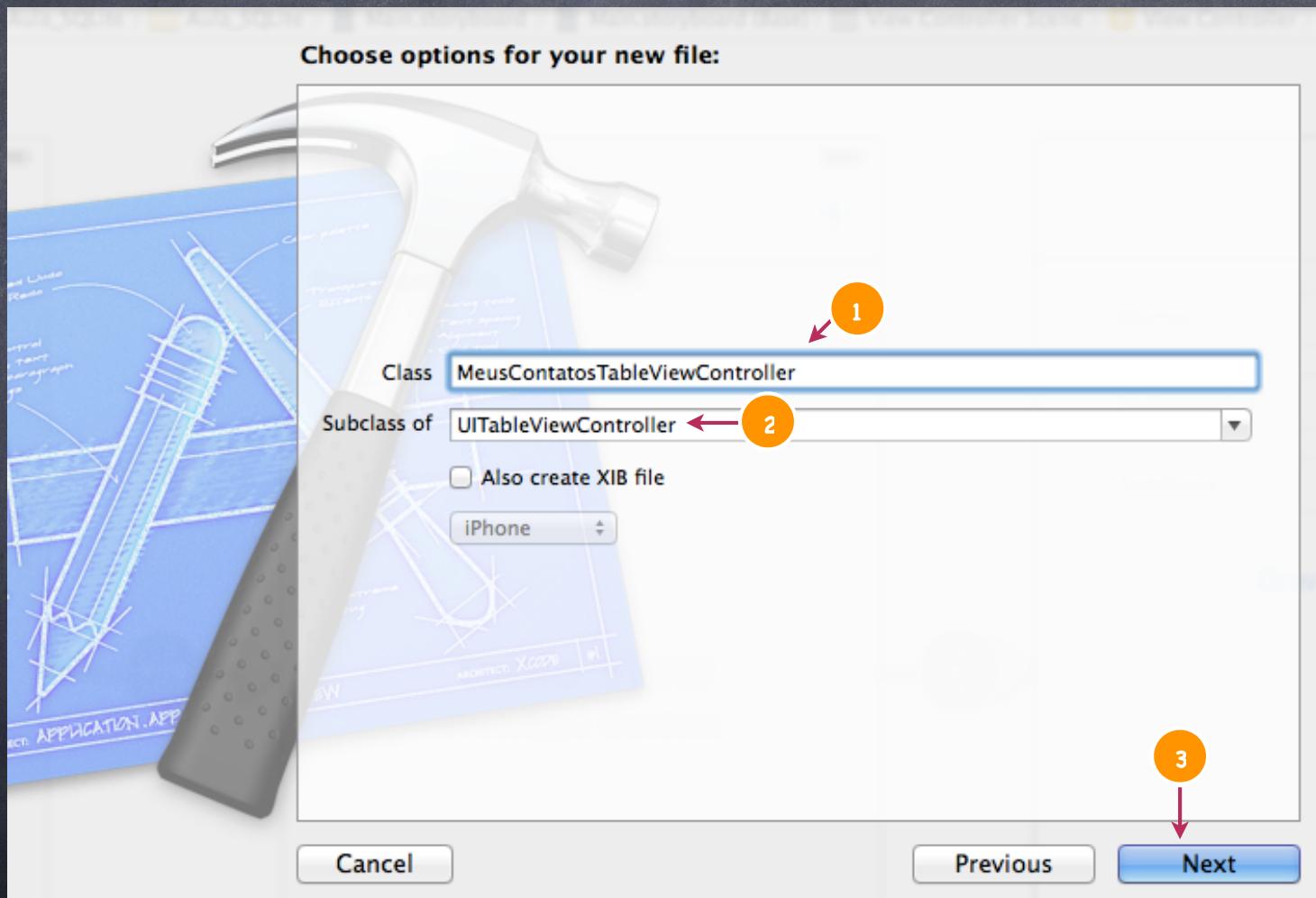
Exibir os Dados

- Vamos criar uma ligação do tipo push (3) da barra de navegação que está no UITableViewController (1) para a View onde gravamos os dados(2).



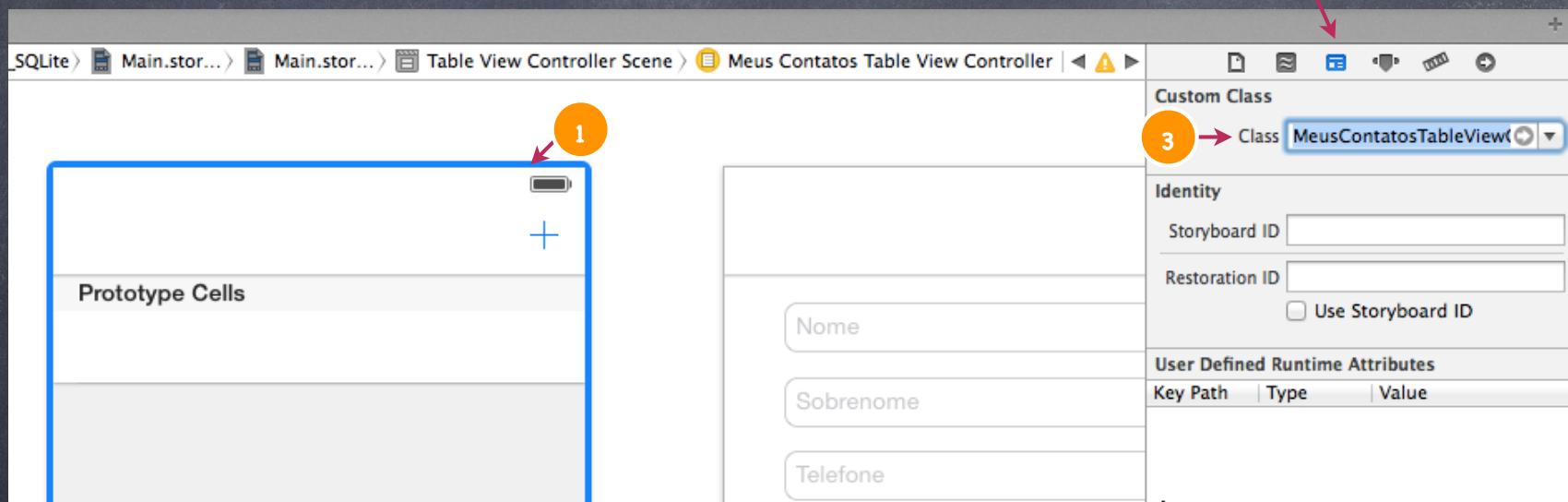
Exibir os Dados

- Vamos criar uma classe chamada MeusContatosTableViewController (1), do tipo UITableViewController (2) para gerenciarmos a exibição dos dados do banco no TableView.



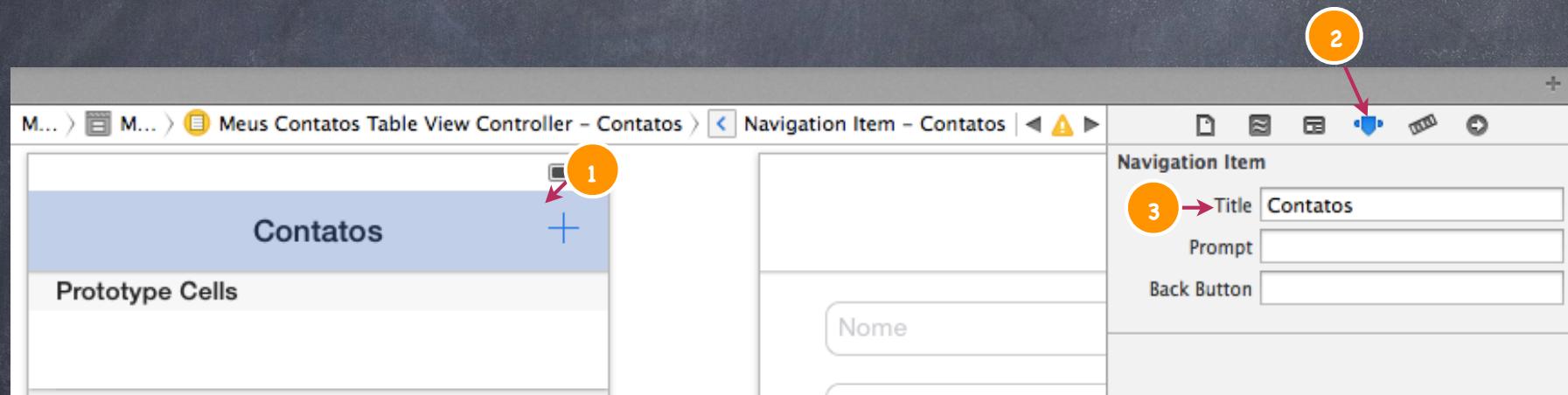
Exibir os Dados

- Selecione o UITableViewController (1) no Identity Inspector (2), escolha a classe que será a dona dessa View, no caso MeusContatosTableViewController(3). Essa será a classe que faremos a programação para que os dados sejam exibidos no TableView.



Exibir os Dados

- Selecione o Navigation (1) no Attributes Inspector (2), altere o Title para Contatos(3), você também pode clicar duas vezes no centro do Navigation e digitar a palavra Contatos.



ContatoBD.h

- Com relação a interface já está tudo pronto, agora precisamos cuidar da implementação para exibir os dados, primeiro crie um método no contatoBD.h para exibir os dados vindos de um select, o método vai possuir o nome: **exibirTodos (1)** e deve retornar um ARRAY mutável.

```
1 // ContatoBD.h
2 // Aula_SQLite
3 //
4 //
5 // Created by agesandro scarpioni on 11/07/14.
6 // Copyright (c) 2014 Agesandro Scarpioni. All rights reserved.
7 //
8 #import <Foundation/Foundation.h>
9 #import "Banco.h"
10 #import "Contato.h"
11
12
13 @interface ContatoBD : Banco{
14 }
15
16 //Retorna o SQL para crair a tabela (deve ser implementado pela sub-classe
17 -(NSString *)getSQLParaCriarTabela;
18
19 //Salva um novo contato ou atualiza um já existente
20 -(void) salvar:(Contato *) contato;
21
22 //retorna um array com o select * do bd.
23 -(NSMutableArray *) exibirTodos ;
24
25 @end
```

ContatoBD.m

- Implemente esse método "exibirTodos" no ContatoBD.m logo abaixo do método "salvar".

```
51 //Seleciona todos contatos cadastrados
52 -(NSMutableArray *) exibirTodos {
53     // Array que vai armazenar a lista de contatos
54     NSMutableArray *contatos =[[NSMutableArray alloc]init];
55     NSString *sql = @"select codigo,nome,sobrenome,telefone from contato order by nome";
56     sqlite3_stmt *stmt;
57     int resultado = sqlite3_prepare_v2(bancoDeDados, [sql UTF8String], -1, &stmt, nil);
58     if (resultado == SQLITE_OK){
59         // enquanto existir contatos
60         while (sqlite3_step(stmt)==SQLITE_ROW) {
61             // CRIA o contato para inserir na lista
62             Contato *c = [Contato alloc];
63             //faz a leitura de cada coluna
64             c.codigo = sqlite3_column_int(stmt,0);
65             char *s = (char *)sqlite3_column_text(stmt, 1);
66             c.nome = s != nil ? [NSString stringWithUTF8String:s]: nil;
67
68             s = (char *)sqlite3_column_text(stmt, 2);
69             c.sobrenome = s != nil ? [NSString stringWithUTF8String:s]: nil;
70             s = (char *)sqlite3_column_text(stmt, 3);
71             c.telefone = s != nil ? [NSString stringWithUTF8String:s]: nil;
72
73             [contatos addObject:c];
74         }
75         sqlite3_finalize(stmt);
76     }
77     return contatos;
78 }
79
80 }
```

Exibir os Dados

- Em MeusContatosTableViewController.h crie uma propriedade do tipo array mutável essa propriedade vai receber a consulta SQL vinda da classe ContatoBD do método exibirTodos.

```
1 //  
2 // MeusContatosTableViewController.h  
3 // Aula_SQLite  
4 //  
5 // Created by agesandro scarpioni on 11/07/14.  
6 // Copyright (c) 2014 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 #import <UIKit/UIKit.h>  
10  
11 @interface MeusContatosTableViewController : UITableViewController  
12 @property (nonatomic, strong) NSMutableArray *contatos;  
13 |  
14  
15 @end  
16
```

Exibir os Dados

- Em MeusContatosTableViewController.m, faça import de ContatosBD.h (1) e o Synthesize de contatos (2), no viewDidLoad descarregue os dados no array (3).

```
1 //  
2 // MeusContatosTableViewController.m  
3 // Aula_SQLite  
4 //  
5 // Created by agesandro scarpioni on 11/07/14.  
6 // Copyright (c) 2014 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
10 #import "MeusContatosTableViewController.h"  
10 #import "ContatoBD.h"  
11  
12 @interface MeusContatosTableViewController ()  
13  
14 @end  
15  
16 @implementation MeusContatosTableViewController  
17 @synthesize contatos; 2  
18  
19  
20 - (id)initWithStyle:(UITableViewStyle)style  
21 {  
22     self = [super initWithStyle:style];  
23     if (self) {  
24         // Custom initialization  
25     }  
26     return self;  
27 }  
28  
29 - (void)viewDidLoad  
30 {  
31     [super viewDidLoad];  
32  
33     ContatoBD *db = [[ContatoBD alloc] init];  
34     [db abrir:@"agenda"]; //abre o banco  
35     contatos = [db exibirTodos];  
36     [db fechar];  
37 }  
38
```

Exibir os Dados

- Ainda em MeusContatosTableViewController.m precisamos apenas gerar as células para exibir os dados no TableView, os métodos já estão disponíveis pois quando criamos a classe informamos que ia ser uma classe filha de UITableViewController, basta implementarmos as linhas abaixo:

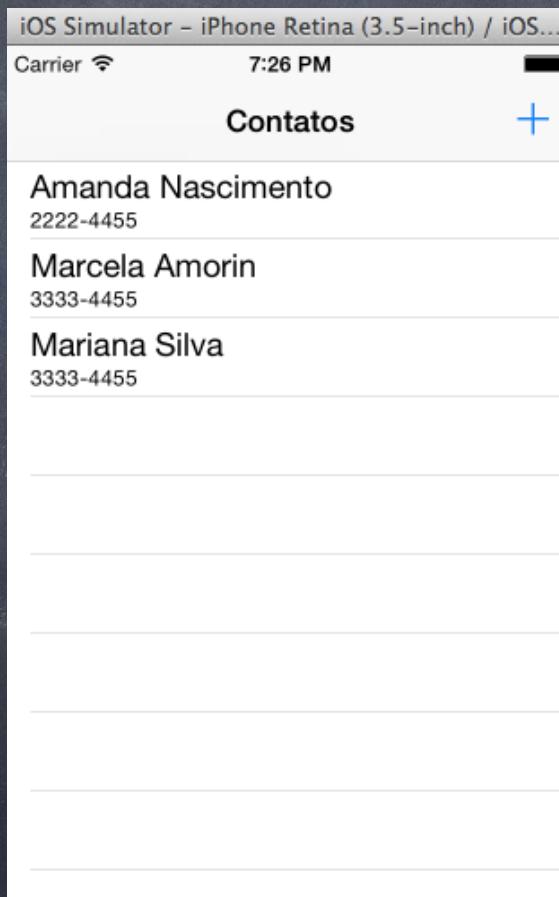


```
44 #pragma mark - Table view data source
45
46 - (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
47 {
48
49     // Return the number of sections.
50     return 1;
51 }
52 }
53
54 - (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
55 {
56     // Return the number of rows in the section.
57     return contatos.count;
58 }
59
60
61 - (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
62 {
63     static NSString *cellIdentifier = @"Cell";
64     UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:cellIdentifier];
65
66
67     // Configure the cell...
68     if (cell==nil){
69         cell=[[UITableViewCell alloc]initWithStyle:UITableViewCellStyleSubtitle reuseIdentifier:cellIdentifier];
70     }
71
72     Contato *contato = [contatos objectAtIndex:indexPath.row ];
73
74     cell.textLabel.text = [NSString stringWithFormat:@"%@ %@", contato.nome , contato.sobrenome];
75     cell.detailTextLabel.text = contato.telefone;
76     return cell;
77 }
78 }
```

The code shows the implementation of a UITableViewDataSource protocol. It includes methods for returning the number of sections (line 51), the number of rows in a section (line 57), and configuring a cell (line 67). Annotations with orange circles and arrows point to these lines: circle 1 points to line 51, circle 2 points to line 57, and circle 3 points to line 74.

Testando

- Execute o aplicativo Command + R, veja que os dados do banco de dados já aparecem na tableView.



App Agenda Persistência

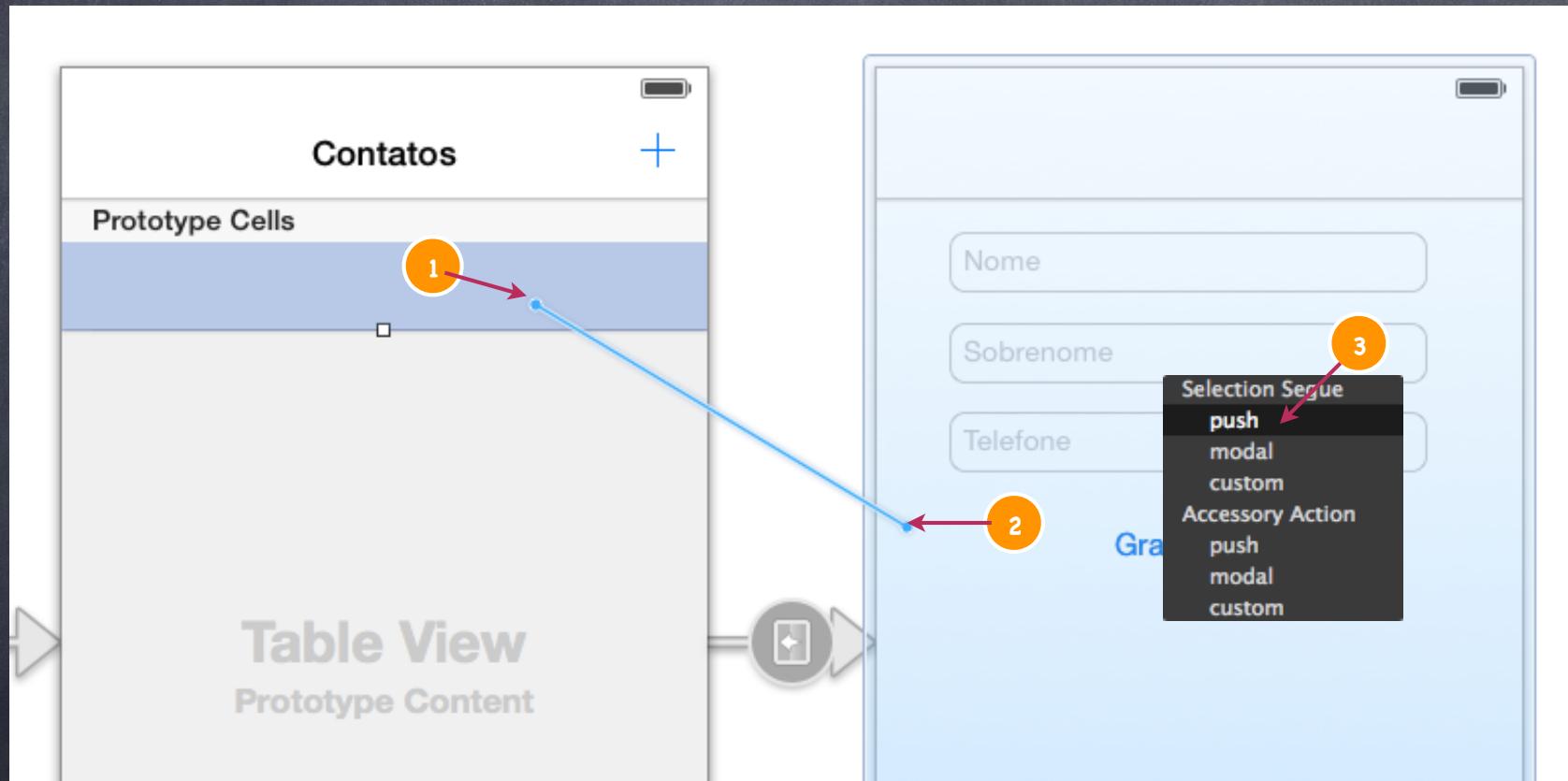
Parte 3 - Editar os dados cadastrados com SQLite

X-Code

Prof. Agesandro Scarpioni

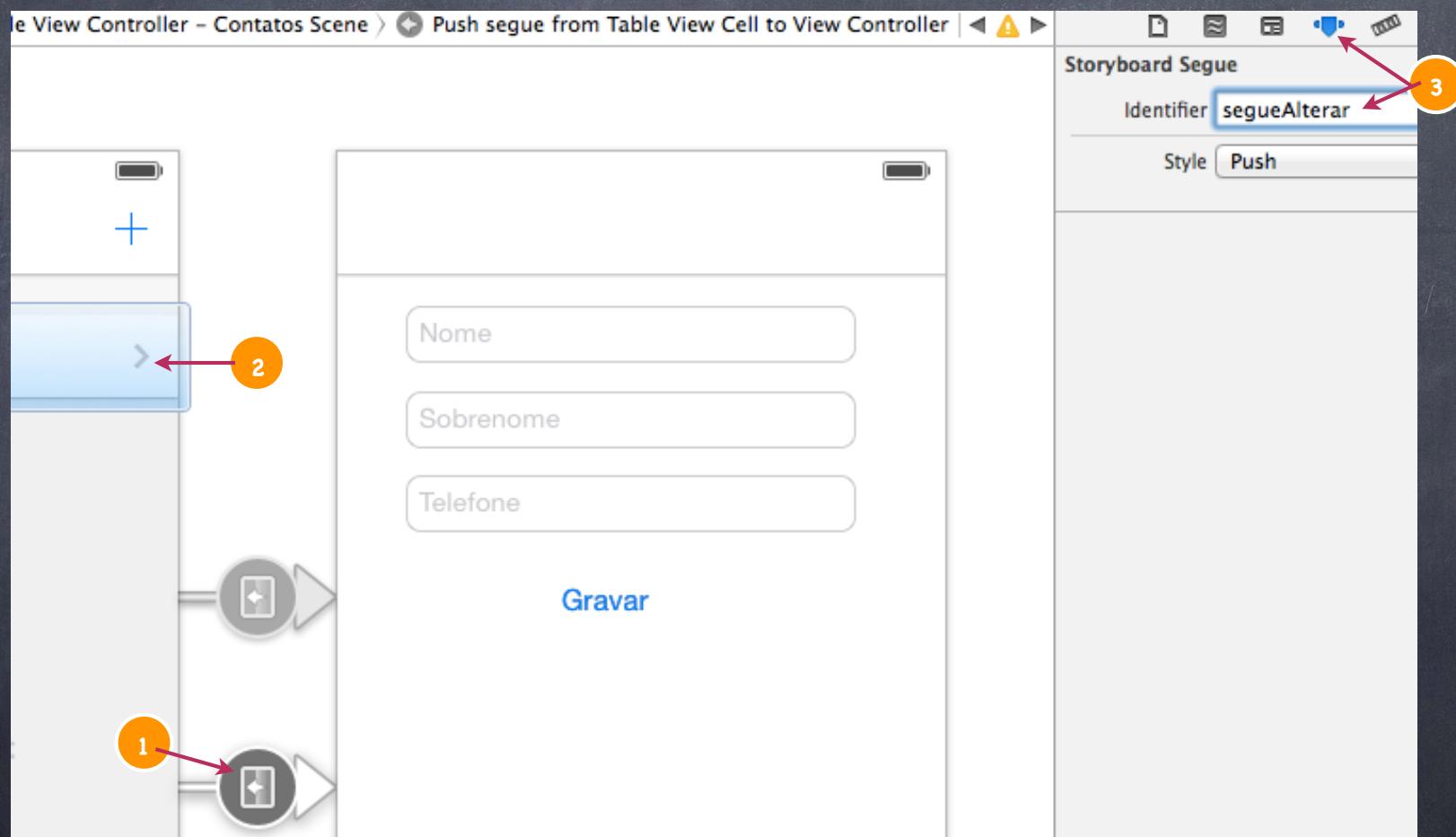
Editar os Dados

- Selecione a célula(1) do TableView e arraste o mouse até a ViewController (2) lado, escolha push(3).



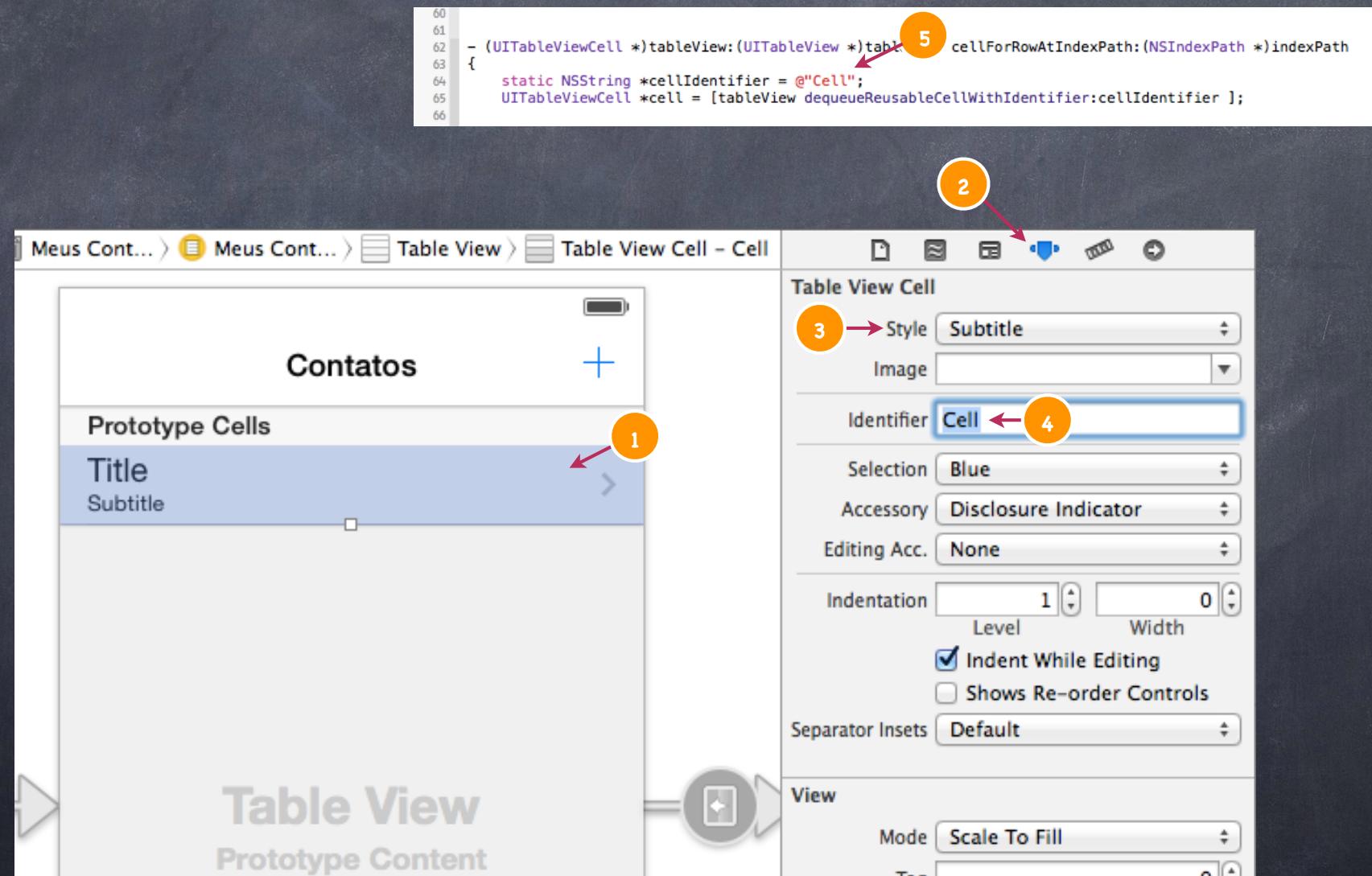
Editar os Dados

- Selecione a Segue que você acabou de criar (1), observe se a Segue é referente a célula(2), altere a propriedade Identifier para: "segueAlterar"(3), dê enter para confirmar.



Core Data

- Selecione a célula (1), em attributes inspector (2) altere o Style (3) para Subtitle e identifique a célula em Identifier (4) com o mesmo nome dado na programação (5).



ViewController.h

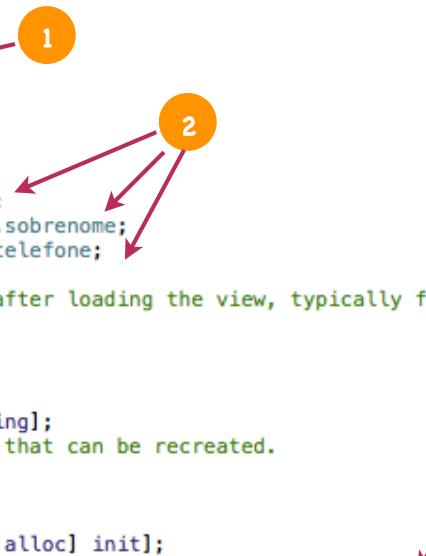
- Faça o import de Contato.h (1) e crie uma propriedade com @property para a classe Contato chamada contato(2).

```
1 //  
2 // ViewController.h  
3 // Aula_SQLite  
4 //  
5 // Created by agesandro scarpioni on 07/07/14.  
6 // Copyright (c) 2014 Agesandro Scarpioni. All rights  
reserved.  
7 //  
8  
9 #import <UIKit/UIKit.h>  
10 #import "Contato.h"  
11  
12 @interface ViewController : UIViewController {  
13  
14     __weak IBOutlet UITextField *txtNome;  
15  
16     __weak IBOutlet UITextField *txtSobrenome;  
17  
18     __weak IBOutlet UITextField *txtTelefone;  
19 }  
20  
21  
22 - (IBAction)Gravar:(id)sender;  
23  
24 @property (nonatomic, retain) Contato *contato;  
25  
26  
27 @end  
28
```

ViewController.m

- Faça o `synthesize` de `contato(1)` no `viewDidLoad` passe os dados da classe contato para as caixas de texto(2), note que não será mais necessário criar o objeto local na linha 37.

```
1 //  
2 // ViewController.m  
3 // Aula_SQLite  
4 //  
5 // Created by agesandro scarpioni on 11/07/14.  
6 // Copyright (c) 2014 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 #import "ViewController.h"  
10 #import "ContatoBD.h"  
11  
12 @interface ViewController()  
13  
14 @end  
15  
16 @implementation ViewController  
17 @synthesize contato;  
18  
19 - (void)viewDidLoad  
20 {  
21     [super viewDidLoad];  
22     txtNome.text = contato.nome;  
23     txtSobrenome.text = contato.sobrenome;  
24     txtTelefone.text = contato.telefone;  
25  
26     // Do any additional setup after loading the view, typically from a nib.  
27 }  
28  
29 - (void)didReceiveMemoryWarning  
30 {  
31     [super didReceiveMemoryWarning];  
32     // Dispose of any resources that can be recreated.  
33 }  
34  
35 - (IBAction)gravar:(id)sender {  
36     ContatoBD *db = [[ContatoBD alloc] init];  
37     Contato *contato = [[Contato alloc] init];  
38     [db abrir:@"agenda"]; //abre o banco  
39     contato.nome = txtNome.text;  
40     contato.sobrenome = txtSobrenome.text;  
41     contato.telefone = txtTelefone.text;  
42     [db salvar:contato];  
43     [db fechar];  
44 }  
45  
46
```



Transportar os Dados

- Vá em MeusContatosTableViewController.m e implemente a passagem dos dados da TableView para a View.

```
78  
79  -(void) prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender{  
80      if ([[segue identifier] isEqualToString:@"segueAlterar"]){  
81          Contato *selectedContato = [contatos objectAtIndex:[[self.tableView indexPathForSelectedRow] row]];  
82          ViewController *destinoViewcontroller = segue.destinationViewController;  
83          destinoViewcontroller.contato =selectedContato;  
84      }  
85 }
```

Ajuste do Gravar

- Pronto, isso já é suficiente para a edição dos dados funcionar, porém, para que seu botão gravar funcione também para insert adicione as linhas do IF (1) e para fechar a view ao clicar em gravar adicione a linha 48.

```
35
36 - (IBAction)gravar:(id)sender {
37     ContatoBD *db = [[ContatoBD alloc] init];
38     [db abrir:@"agenda"]; //abre o banco
39     if(!contato){
40         contato = [[Contato alloc]init];
41     }
42     contato.nome = txtNome.text;
43     contato.sobrenome = txtSobrenome.text;
44     contato.telefone = txtTelefone.text;
45
46     [db salvar:contato];
47     [db fechar];
48     [self.navigationController popViewControllerAnimated:YES];
49 }
50 @end
51
```



OBS: Lembre-se que retiramos no slide 57 a linha que criava uma instância de Contato, por isso precisamos do IF para criarmos a instância quando o contato não vier carregado da tableView na tela anterior.

Atualizar UITableView

FIAP

- Em MeusContatosTableViewController.m crie o método `viewDidAppear(1)` para que ao clicarmos no gravar a `tableView` seja atualizada ao fazer `update`, **retire** as linhas do `viewDidLoad(2)` e passe para o `viewDidAppear(3)`, dessa forma a `tableView` também será atualizada no `insert`.

```
1 //  
2 // MeusContatosTableViewController.m  
3 // Aula_SQLite  
4 //  
5 // Created by agesandro scarpioni on 11/07/14.  
6 // Copyright (c) 2014 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 #import "MeusContatosTableViewController.h"  
10 #import "ContatoBD.h"  
11  
12 @interface MeusContatosTableViewController ()  
13  
14 @end  
15  
16 @implementation MeusContatosTableViewController  
17 @synthesize contatos;  
18  
19  
20 - (id)initWithStyle:(UITableViewStyle)style  
21 {  
22     self = [super initWithStyle:style];  
23     if (self) {  
24         // Custom initialization  
25     }  
26     return self;  
27 }  
28  
29 - (void)viewDidLoad  
30 {  
31     [super viewDidLoad];  
32  
33     ContatoBD *db = [[ContatoBD alloc] init];  
34     [db abrir:@"agenda"]; //abre o banco  
35     contatos = [db exibirTodos];  
36     [db fechar];  
37 }
```

```
1 39 -(void)viewDidAppear:(BOOL)animated{  
2 40     [super viewDidAppear:animated];  
3 41     [self.tableView reloadData];  
4 42 }  
  
32 }  
33 -(void)viewDidAppear:(BOOL)animated{  
34     [super viewDidAppear:animated];  
35     ContatoBD *db = [[ContatoBD alloc] init];  
36     [db abrir:@"agenda"]; //abre o banco  
37     contatos = [db exibirTodos];  
38     [db fechar];  
39     [self.tableView reloadData];  
40 }  
41 }  
42 }  
43 }
```

Testando

• Agora vamos testar a edição

The figure consists of three side-by-side screenshots of an iPhone's Contacts application, showing the process of editing a contact.

Screenshot 1 (Left): Shows the main Contacts screen at 3:07 AM. A contact named "Sueli Cruz" is selected, indicated by a red oval. Her details are shown: Name "Sueli Cruz" and Phone number "7676-6054".

Screenshot 2 (Middle): Shows the contact details screen for "Sueli Cruz" at 3:09 AM. The name field contains "Crupie" (circled in red), and the phone number field contains "7676-6054". A red arrow points from the "Sueli Cruz" entry in Screenshot 1 to the "Crupie" entry here. Another red arrow points from the "Gravar" (Save) button on the keyboard to the "Crupie" entry. The keyboard is visible at the bottom.

Screenshot 3 (Right): Shows the main Contacts screen at 3:09 AM after saving. The contact "Sueli Cruz" has been updated to "Sueli Crupie", which is circled in red. Her phone number remains "7676-6054".

App Agenda Persistência

Parte 4 - Deletar os dados cadastrados com SQLite

X-Code

Prof. Agesandro Scarpioni

Deletar Dados

- No ContatoBD.h, declare o método deletar da linha 26.

```
1 //  
2 // ContatoBD.h  
3 // Aula_SQLite  
4 //  
5 // Created by agesandro scarpioni on 11/07/14.  
6 // Copyright (c) 2014 Agesandro Scarpioni. All rights reserved.  
7 //  
8 #import <Foundation/Foundation.h>  
9 #import "Banco.h"  
10 #import "Contato.h"  
11  
12  
13 @interface ContatoBD : Banco{  
14 }  
15 //Retorna o SQL para cair a tabela (deve ser implementado pela sub-classe  
16 -(NSString *)getSQLParaCriarTabela;  
17  
18 //Salva um novo contato ou atualiza um já existente  
19 -(void) salvar:(Contato *) contato;  
20  
21 //retorna um array com o select * do bd.  
22 -(NSMutableArray *) exibirTodos ;  
23  
24 //Deletea um contato  
25 -(void) deletar:(Contato *) contato; ←—————  
26  
27  
28  
29 @end  
30
```

Deletar Dados

- No ContatoBD.m, traga a declaração do deletar da classe contatoBD.h e faça implementação abaixo onde registro é excluído pelo código, no próximo slide existe um exemplo de delete pelo nome aproximado (like), note a diferença entre um parâmetro com campo numérico e um parâmetro string.

```
52
53 //Delete um contato
54 -(void) deletar:(Contato *) contato {
55     char *sql = "delete from contato where codigo=?";
56     sqlite3_stmt *stmt;
57     int resultado = sqlite3_prepare_v2(bancoDeDados, sql, -1, &stmt, nil);
58     if (resultado == SQLITE_OK){
59         //informando o id para apagar
60         sqlite3_bind_int(stmt,1,contato.codigo);
61         //executa o sql
62         resultado = sqlite3_step(stmt);
63         if (resultado == SQLITE_DONE){
64             //DELETADO COM SUCESSO
65         }
66         sqlite3_finalize(stmt);
67     }
68 }
69 }
```

Classe ContatoBD

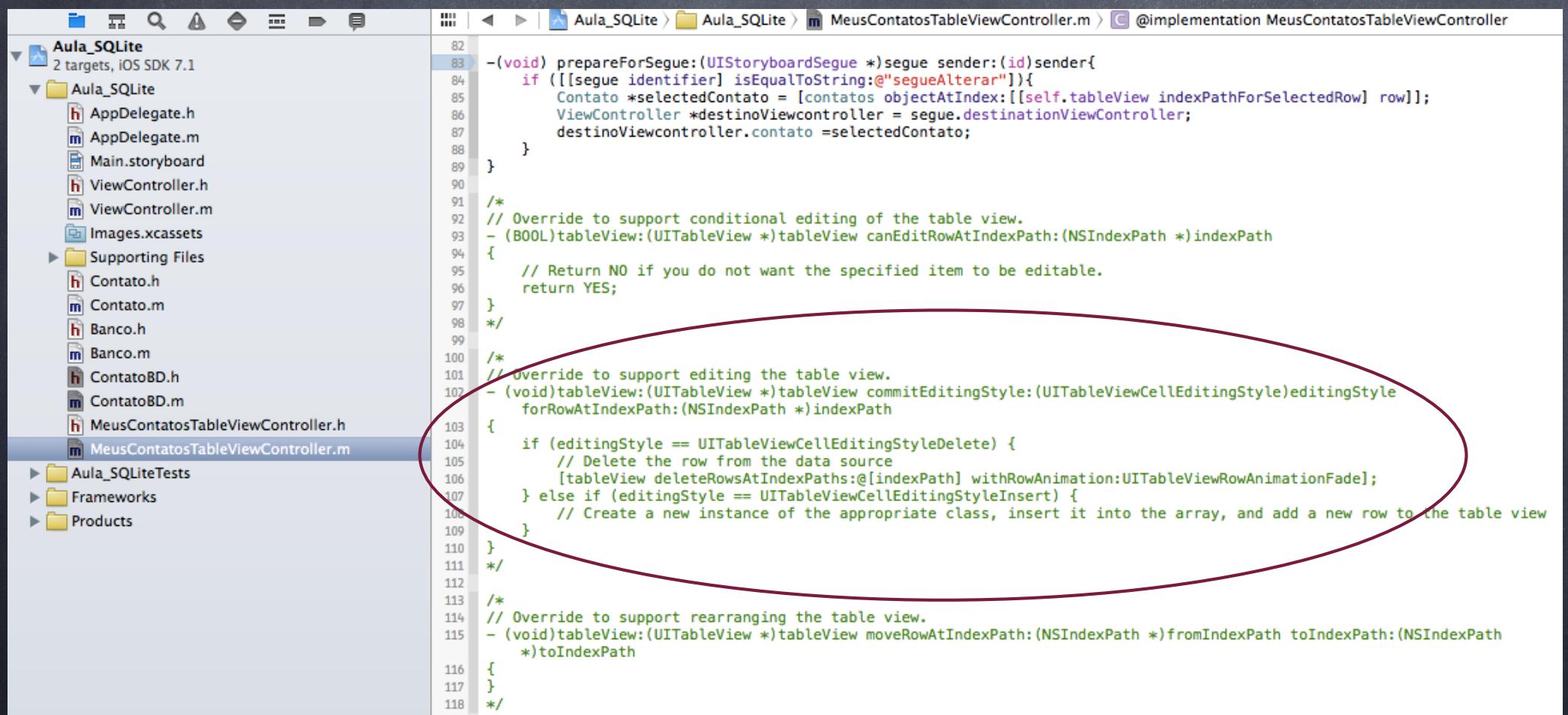
- Exemplo com delete pelo nome aproximado like, usado para apagar todos os contatos que iniciam com uma certa letra, note a diferença entre um parâmetro com campo numérico do slide anterior e um parâmetro string desse slide.

NÃO DIGITAR ESSE CÓDIGO APENAS COMPARE

```
69  
70 //Deleta todos os contatos que iniciam com uma letra  
71 -(void) deletarTodos:(NSString *) letra {  
72     char *sql = "delete from contato where nome like ?";  
73     sqlite3_stmt *stmt;  
74     int resultado = sqlite3_prepare_v2(bancoDeDados, sql, -1, &stmt, nil);  
75     if (resultado == SQLITE_OK){  
76         //informando o nome para apagar  
77         sqlite3_bind_text(stmt,1, [letra UTF8String],-1,nil);  
78  
79         //executa o sql  
80         resultado = sqlite3_step(stmt);  
81         if (resultado == SQLITE_DONE){  
82             //DELETADO COM SUCESSO  
83         }  
84         sqlite3_finalize(stmt);  
85     }  
86 }  
87 }
```

Deletar Dados

- Procure em `MeusContatosTableViewController.m` o método `CommitEditingStyle` e retire os comentários.



The screenshot shows the Xcode interface with the project "Aula_SQLite" selected. The file "MeusContatosTableViewController.m" is open in the editor. A red oval highlights the code block for the `commitEditingStyle:` method, which is used for handling row deletion in the UITableView.

```
82
83  -(void) prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender{
84      if ([[segue identifier] isEqualToString:@"segueAlterar"]){
85          Contato *selectedContato = [contatos objectAtIndex:[[self.tableView indexPathForSelectedRow] row]];
86          ViewController *destinoViewController = segue.destinationViewController;
87          destinoViewController.contato =selectedContato;
88      }
89  }
90
91 /*
92 // Override to support conditional editing of the table view.
93 - (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath *)indexPath
94 {
95     // Return NO if you do not want the specified item to be editable.
96     return YES;
97 }
98 */
99 /*
100 // override to support editing the table view.
101 - (void)tableView:(UITableView *)tableView commitEditingStyle:(UITableViewCellEditingStyle)editingStyle
102 forRowAtIndexPath:(NSIndexPath *)indexPath
103 {
104     if (editingStyle == UITableViewCellEditingStyleDelete) {
105         // Delete the row from the data source
106         [tableView deleteRowsAtIndexPaths:@[indexPath] withRowAnimation:UITableViewRowAnimationFade];
107     } else if (editingStyle == UITableViewCellEditingStyleInsert) {
108         // Create a new instance of the appropriate class, insert it into the array, and add a new row to the table view
109     }
110 }
111 */
112 /*
113 // Override to support rearranging the table view.
114 - (void)tableView:(UITableView *)tableView moveRowAtIndexPath:(NSIndexPath *)fromIndexPath toIndexPath:(NSIndexPath *)
115 *toIndexPath
116 {
117 }
118 */
119
```

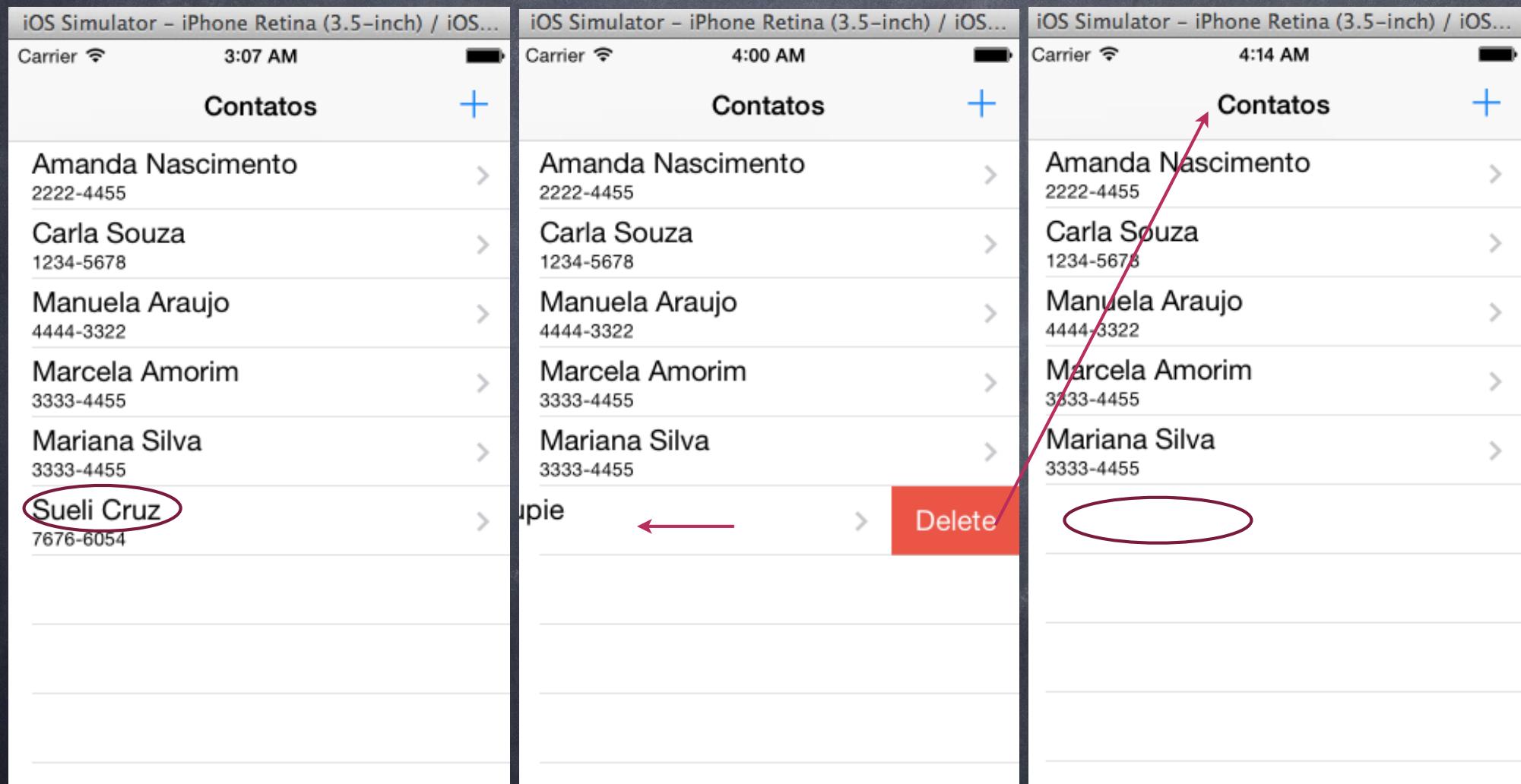
Deletar Dados

- Ao retirar os comentários implemente apenas a primeira parte do IF.

```
100 // Override to support editing the table view.
101 - (void)tableView:(UITableView *)tableView commitEditingStyle:(UITableViewCellEditingStyle)editingStyle
102     forRowAtIndexPath:(NSIndexPath *)indexPath
103 {
104     if (editingStyle == UITableViewCellEditingStyleDelete) {
105         // Delete the row from the data source
106         ContatoBD *db = [[ContatoBD alloc] init];
107         [db abrir:@"agenda"]; //abre o b;anco
108         Contato *selectedContato = [contatos objectAtIndex:indexPath.row];
109         [db deletar:selectedContato];
110         [db fechar];
111         [contatos removeObjectAtIndex:indexPath.row];
112         [tableView deleteRowsAtIndexPaths:@[indexPath] withRowAnimation:UITableViewRowAnimationFade];
113
114     } else if (editingStyle == UITableViewCellEditingStyleInsert) {
115         // Create a new instance of the appropriate class, insert it into the array, and add a new row to the table view
116     }
117 }
118 }
119 }
```

Testando

• Agora vamos testar a exclusão de um registro.



Prática

- Altere seu programa para quando clicarmos no botão (+) apareça no título da View a palavra CADASTRAR e quando clicarmos na célula da tableView apareça no título da View a palavra EDITAR.
- Inclua na TableView um Activity para ser exibido enquanto os dados vindos da View são atualizados, depois de atualizados esconda o Activity.