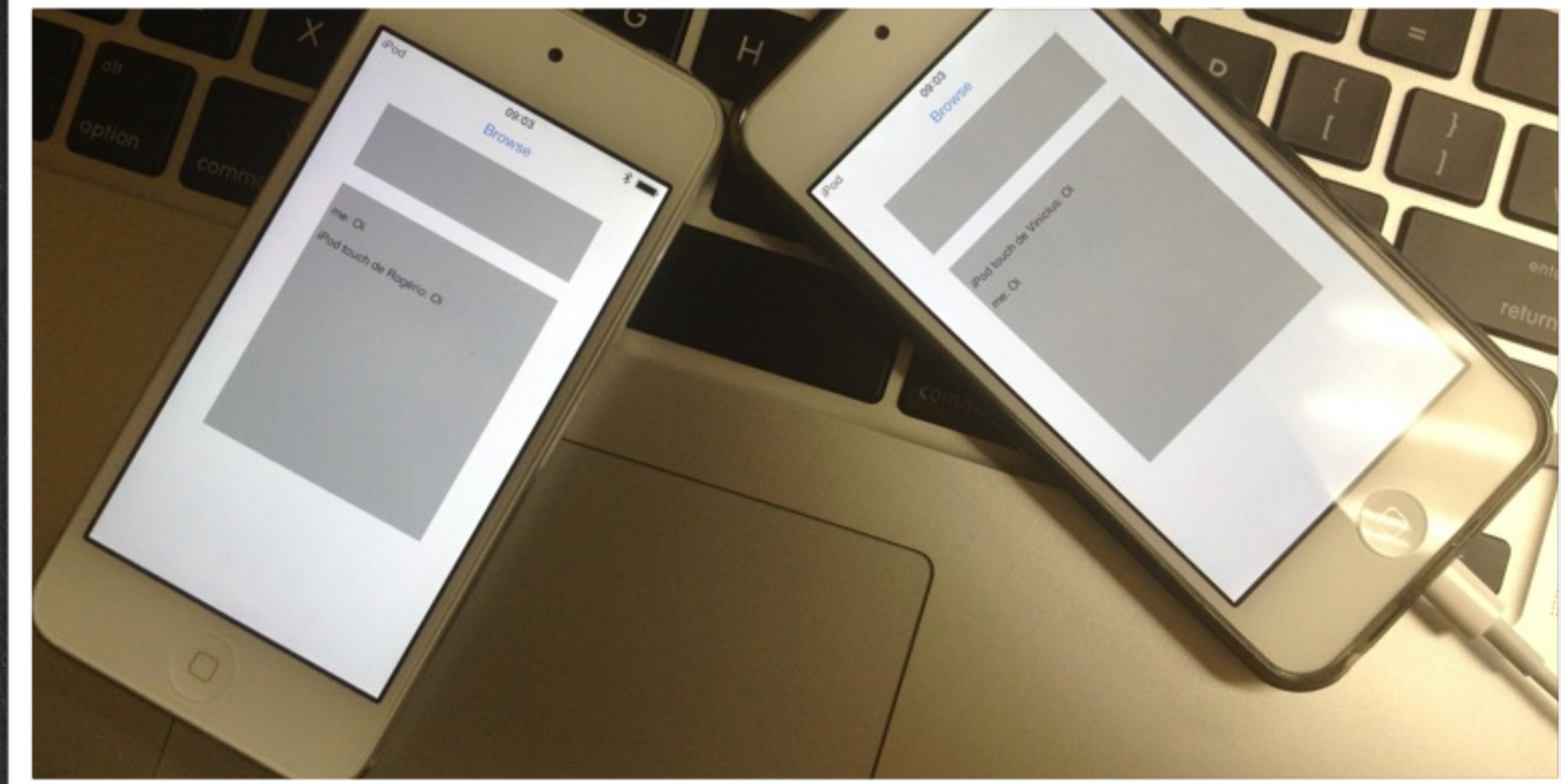


Multipeer Connectivity

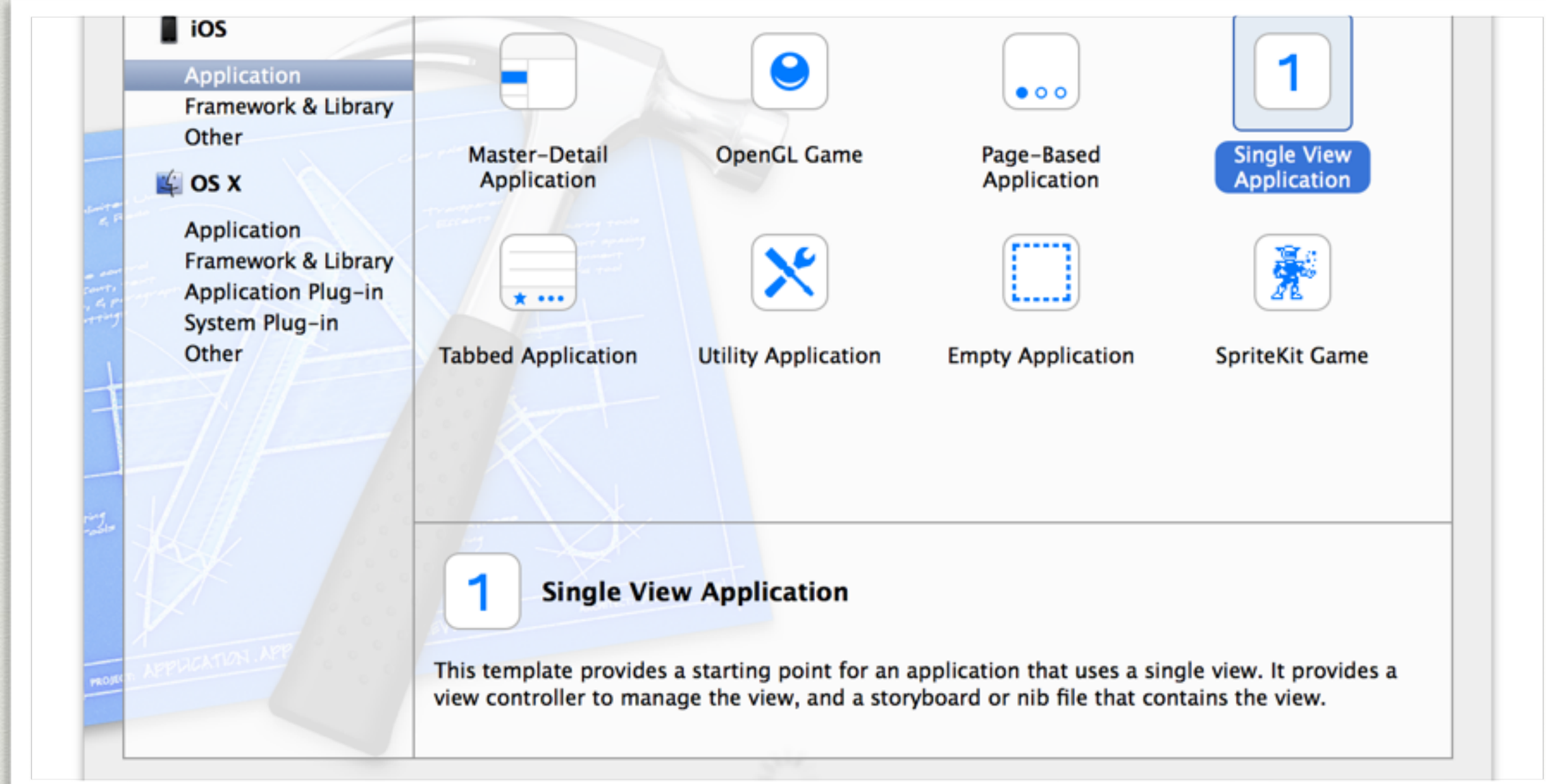
Rogério Bordignon e Vinicius Lima

Introdução

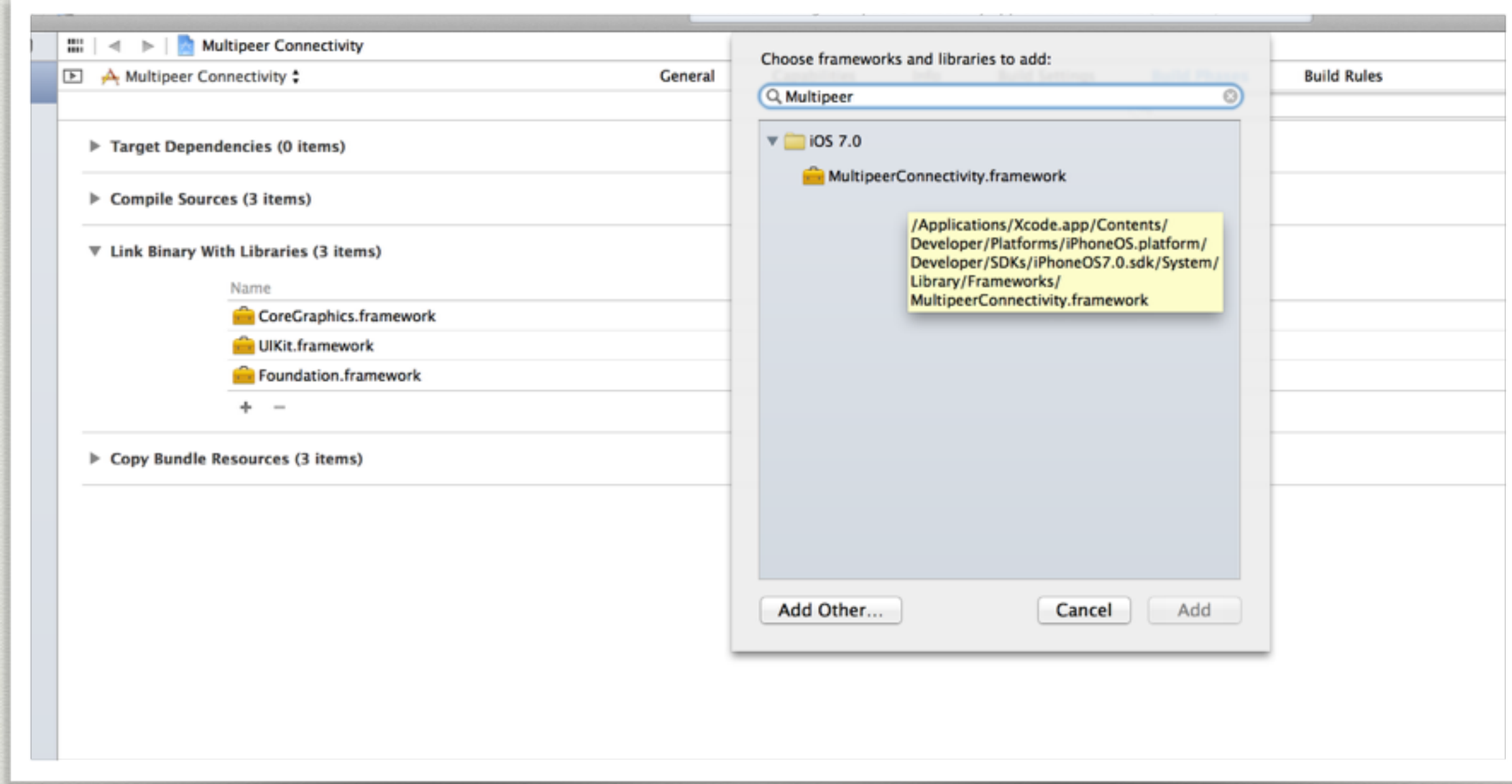
Multipeer Connectivity é um framework introduzido no iOS 7, no lugar do GKPeerPickerController, que oferece suporte para encontrar devices próximos usando redes Wi-Fi e Bluetooth assim como a comunicação, compartilhamento de dados e recursos entre os mesmos.

Criando o Código

Nesse tutorial criaremos um pequeno chat entre dois ou mais dispositivos.



Crie um projeto no xCode - File > New > Project do tipo (Single View Application)



Adicione o Framework Multipeer Connectivity framework.
Click no projeto > Build Phases > Link Binary With Libraries.

```
#import <UIKit/UIKit.h>
#import <MultipeerConnectivity/MultipeerConnectivity.h>

@interface ViewController : UIViewController <MCBrowserViewControllerDelegate, MCSessionDelegate, UITextFieldDelegate>


@property (nonatomic, strong) UIButton *browserButton;
@property (nonatomic, strong) UITextField *chatBox;
@property (nonatomic, strong) UITextView *textBox;

@end
```

Importe a Biblioteca do Multipeer

Adicione os seguintes delegates no seu .h

<MCBrowserViewControllerDelegate, MCSessionDelegate, UITextFieldDelegate>

Adicione as variáveis igual a imagem

UIButton, UITextField e UITextView

- Depois dessa etapa irão aparecer alguns warnings devido a falta de alguns métodos necessários para esses delegates.

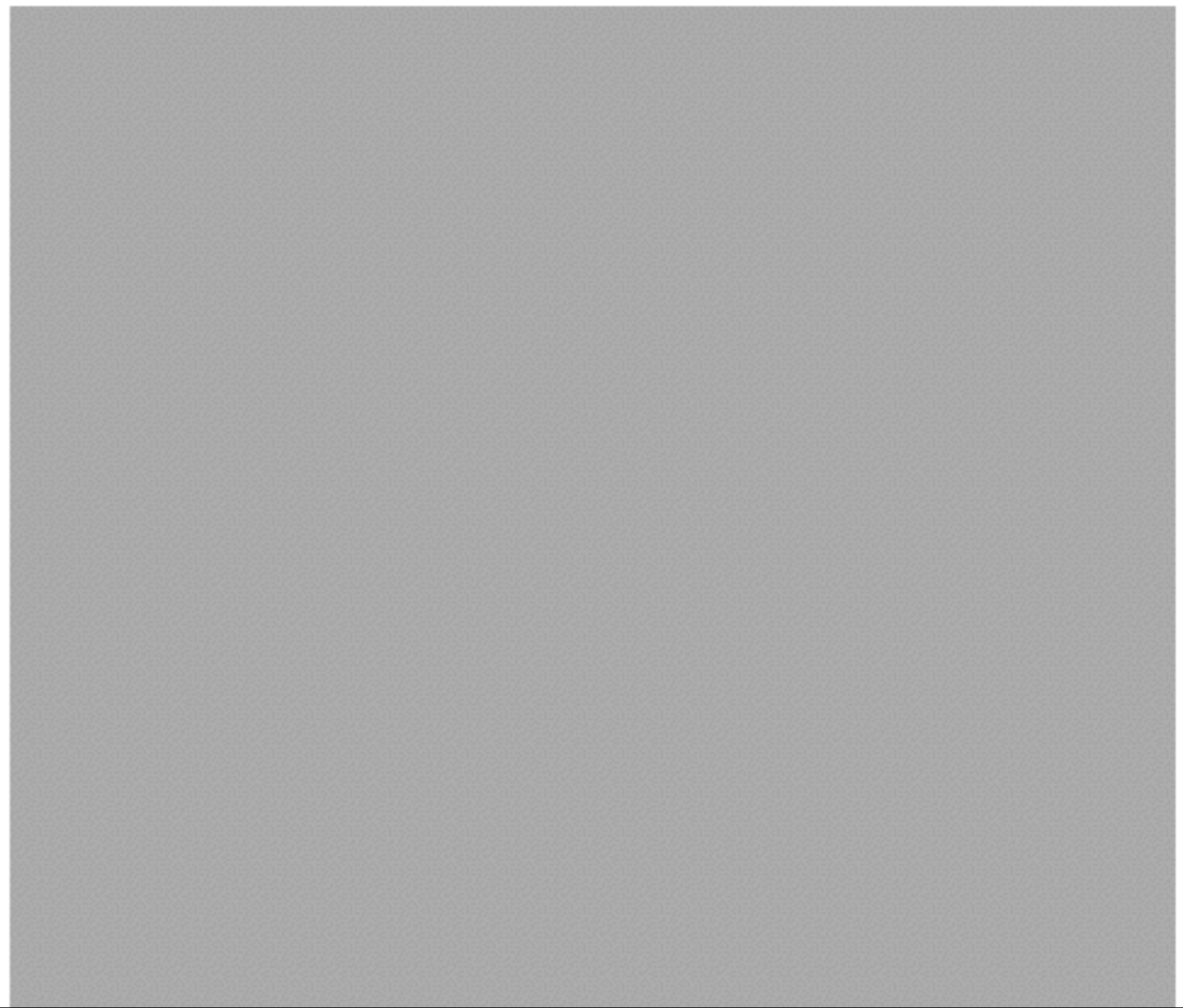
- (Resolveremos isto mais pra frente)

```objc
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
    [self setUpUI];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (void) setUpUI{
    //  Setup the browse button
    self.browserButton = [UIButton buttonWithType:UIButtonTypeSystem];
    [self.browserButton setTitle:@"Browse" forState:UIControlStateNormal];
    self.browserButton.frame = CGRectMake(130, 20, 60, 30);
    [self.view addSubview:self.browserButton];
    [self.browserButton addTarget:self action:@selector(showBrowserVC) forControlEvents:UIControlEventTouchUpInside];

    //  Setup TextBox
    self.textBox = [[UITextView alloc] initWithFrame: CGRectMake(40, 150, 240, 270)];
    self.textBox.editable = NO;
    self.textBox.backgroundColor = [UIColor lightGrayColor];
    [self.view addSubview: self.textBox];

    //  Setup ChatBox
    self.chatBox = [[UITextField alloc] initWithFrame: CGRectMake(40, 60, 240, 70)];
    self.chatBox.backgroundColor = [UIColor lightGrayColor];
    self.chatBox.returnKeyType = UIReturnKeySend;
    self.chatBox.delegate = self;
    [self.view addSubview:self.chatBox];
}
```

Desenvolveremos então, o botão e os textos em um método separado e depois adicionaremos o mesmo no viewDidLoad (O Código dentro do método setUpUI pode ser jogado direto no viewDidLoad)

Carrier 9:00 AM

Browse



```
//

#import <UIKit/UIKit.h>
#import <MultipeerConnectivity/MultipeerConnectivity.h>

@interface ViewController : UIViewController <MCBrowserViewControllerDelegate, MCSessionDelegate, UITextFieldDelegate>


@property (nonatomic, strong) MCBrowserViewController *browserVC;
@property (nonatomic, strong) MCAdvertiserAssistant *advertiser;
@property (nonatomic, strong) MCSession *mySession;
@property (nonatomic, strong) MCPeerID *myPeerID;

@property (nonatomic, strong) UIButton *browserButton;
@property (nonatomic, strong) UITextField *chatBox;
@property (nonatomic, strong) UITextView *textBox;

@end
```

Iremos agora adicionar as variáveis responsáveis pela conexão. MCPeerID, MCSession, MCAdvertiserAssistant e MCBrowserViewController.

- MCPeerID representa um ponto na sessão Multipeer.
- Um objeto MCSession permite e gerencia a comunicação entre todos os pares em uma sessão Multipeer.
- MCAdvertiserAssistant apresenta convites de entrada para o usuário e processa respostas dos usuários.
- A classe MCBrowserViewController apresenta dispositivos próximos ao usuário e permite que o usuário convide dispositivos próximos para uma sessão.

```
(void)viewDidLoad

  [super viewDidLoad];
  // Do any additional setup after loading the view, typically from a nib.
  [self setUpUI];
  [self setUpMultipeer];


(void)didReceiveMemoryWarning {...}

(void) setUpUI {...}

(void) setUpMultipeer {
  //  Setup peer ID
  self.myPeerID = [[MCPeerID alloc] initWithDisplayName:[UIDevice currentDevice].name];

  //  Setup session
  self.mySession = [[MCSession alloc] initWithPeer:self.myPeerID];
  self.mySession.delegate = self;

  //  Setup BrowserViewController
  self.browserVC = [[MCBrowserViewController alloc] initWithServiceType:@"chat" session:self.mySession];
  self.browserVC.delegate = self;

  //  Setup Advertiser
  self.advertiser = [[MCAdvertiserAssistant alloc] initWithServiceType:@"chat" discoveryInfo:nil session:self.mySessi
  [self.advertiser start];


(void) showBrowserVC{
```

Agora, iremos desenvolver as variáveis em um novo método e depois adiciona-las no viewDidLoad.

- Ao inicializarmos o MyPeerID colocamos para para ele vir com o nome de cada device. (Iphone de Fulano).

- Em MySession usamos o MyPeerID e colocamos o delegate em self.

- O browserVC por sua vez, usa o MySession.

```
    self.textBox.text = [self.textBox.text stringByAppendingString:finalText];
}

#pragma marks MCBrowserViewControllerDelegate

 // Notifies the delegate, when the user taps the done button
- (void)browserViewControllerDidFinish:(MCBrowserViewController *)browserViewController{
    [self dismissBrowserVC];
}

// Notifies delegate that the user taps the cancel button.
- (void)browserViewControllerWasCancelled:(MCBrowserViewController *)browserViewController{
    [self dismissBrowserVC];
}

#pragma marks UITextFieldDelegate

- (BOOL)textFieldShouldReturn:(UITextField *)textField{
    [textField resignFirstResponder];
    [self sendText];
    return YES;
}

#pragma marks MCSessionDelegate
// Remote peer changed state
```

Criaremos os método responsáveis pelos delegates.
Primeiro para o MCBrowserViewControllerDelegate e para o UITextFieldDelegate

- O delegate browserViewController constrói a view onde iremos procurar devices, ela já vem pré definida. Iremos inicialmente apenas fazer com que ela apareça, conecte as pessoas e volte. Esses delegates funcionam adicionando o botão de Cancel e Done dessa view.

Carrier 8:56 AM

Cancel Done

NEARBY

Searching...

```objc
}

#pragma marks MCSessionDelegate
// Remote peer changed state
- (void)session:(MCSession *)session peer:(MCPeerID *)peerID didChangeState:(MCSessionState)state{

}

// Received data from remote peer
- (void)session:(MCSession *)session didReceiveData:(NSData *)data fromPeer:(MCPeerID *)peerID{
    //  Decode data back to NSString
    NSString *message = [[NSString alloc] initWithData:data encoding:NSUTF8StringEncoding];

    //  append message to text box:
    dispatch_async(dispatch_get_main_queue(), ^{
        [self receiveMessage:message fromPeer:peerID];
    });
}

// Received a byte stream from remote peer
- (void)session:(MCSession *)session
didReceiveStream:(NSInputStream *)stream
       withName:(NSString *)streamName
       fromPeer:(MCPeerID *)peerID{

}

// Start receiving a resource from remote peer
```

Agora os delegates relacionados
à conectividade

```
        fromPeer:(MCPeerID *)peerID{

}

// Start receiving a resource from remote peer
- (void)session:(MCSession *)session
didStartReceivingResourceWithName:(NSString *)resourceName
        fromPeer:(MCPeerID *)peerID
    withProgress:(NSProgress *)progress{

}

// Finished receiving a resource from remote peer and saved the content in a temporary locat
// the app is responsible for moving the file to a permanent location within its sandbox
- (void)session:(MCSession *)session
didFinishReceivingResourceWithName:(NSString *)resourceName
        fromPeer:(MCPeerID *)peerID
           atURL:(NSURL *)localURL
       withError:(NSError *)error{

}


@end
```

Mesmos as funções sem corpo, ainda são necessárias para o aplicativo funcionar

- Os métodos estão com sua funcionalidade comentada antes da sua assinatura no .m

```
}

- (void) showBrowserVC {
    [self presentViewController:self.browserVC animated:YES completion:nil];
}

- (void) dismissBrowserVC {
    [self.browserVC dismissViewControllerAnimated:YES completion:nil];
}

- (void) sendText {
    //  Retrieve text from chat box and clear chat box
    NSString *message = self.chatBox.text;
    self.chatBox.text = @"";

    //  Convert text to NSData
    NSData *data = [message dataUsingEncoding:NSUTF8StringEncoding];

    //  Send data to connected peers
    NSError *error;
    [self.mySession sendData:data toPeers:[self.mySession connectedPeers] withMode:MCSessionSendDataUnreliable error:&error]

    //  Append your own message to text box
    [self receiveMessage: message fromPeer: self.myPeerID];
}

- (void) receiveMessage: (NSString *) message fromPeer: (MCPeerID *) peer{ ... }

#pragma marks MCBrowserViewControllerDelegate

 // Notifies the delegate, when the user taps the done button
- (void)browserViewControllerDidFinish:(MCBrowserViewController *)browserViewController{
    [self dismissBrowserVC];
```

Adicionaremos os métodos finais: sendText
o qual transforma sua mensagem em data e envia para todos os devices emparelhados.

```
    [self receiveMessage: message fromPeer: self.myPeerID];
}

- (void) receiveMessage: (NSString *) message fromPeer: (MCPeerID *) peer{
    //  Create the final text to append
    NSString *finalText;
    if (peer == self.myPeerID) {
        finalText = [NSString stringWithFormat:@"\nme: %@ \n", message];
    }
    else{
        finalText = [NSString stringWithFormat:@"\n%@: %@ \n", peer.displayName, message];
    }

    //  Append text to text box
    self.textBox.text = [self.textBox.text stringByAppendingString:finalText];
}

#pragma marks MCBrowserViewControllerDelegate

 // Notifies the delegate, when the user taps the done button
- (void)browserViewControllerDidFinish:(MCBrowserViewController *)browserViewController{
    [self dismissBrowserVC];
}

// Notifies delegate that the user taps the cancel button.
```

No método receiveMessage existe um if, que troca o nome da pessoa que fala no chat.

- Algum problema? O código todo se encontra no github: https://github.com/MackMobile/ios-demo-multipeer

http://www.appcoda.com/intro-ios-multipeer-connectivity-programming/

http://nshipster.com/multipeer-connectivity/

http://techmaster.vn/2013/09/multipeer-connectivity-quick-tutorial/

Referências

Este tutorial é uma tradução do site: http://techmaster.vn/2013/09/multipeer-connectivity-quick-tutorial/