04

Componente de mensagem do iOS

Transcrição

[00:00] Agora é hora de implementarmos o componente de mensagem. Nós acabamos de criar o action sheet, que é esse menu que é disparado quando nós fazemos o long press na célula e nós temos um botão “enviar SMS”, então quando eu clicar aqui o que deve acontecer é abrir o componente de mensagem nativo do iOS, pro usuário conseguir enviar as mensagens para os alunos.

[00:22] O primeiro passo é criarmos uma classe que implemente esses métodos desse componente de mensagem. Aqui onde está “Componentes” eu vou criar uma nova classe, botão direito, new file, “Cocoa Touch Class” mesmo, vou dar um next e aqui eu vou chamá-la de “Mensagem”. Next e create.

[00:46] Como nós vamos precisar utilizar o framework de mensagem do iOS, nós vamos ter que importá-lo para conseguir utilizar os seus protocolos e os seus métodos. O primeiro passo é importar esse componente. Vou dar um “import”, ele se chama “MessageUI”, é esse primeiro aqui mesmo, esse que nós vamos utilizar.

[01:11] Quando nós trabalhamos com ele, temos que implementar um método de delegate. Pra isso vamos utilizar o seu protocolo, que é esse “MFMessageComposeViewControllerDelegate”, é esse aqui. Quando eu utilizo esse protocolo, ele vai dar um erro que eu preciso implementar esse método. Então aqui nós já podemos utilizar esse método.

[01:32] Vou criar aqui um mark pra deixar os métodos de delegate, “MARK: MessageComposeDelegate” e aqui eu implemento. Qual é o método que eu vou utilizar? É esse “didFinishWith result”. O que eu vou querer fazer quando ele terminar de operar esse componente? Eu vou simplesmente desaparecer com ele, então eu vou chamar aqui o “controller.dismiss”, animado eu passo verdadeiro e aqui em “completion” eu passo “nil”. Implementei o método que eu preciso pra esse componente funcionar.

[02:09] Agora eu preciso criar um método para montar esse componente, ou seja, um método que devolva esse componente pra nós utilizarmos no Home Table View Controller. Então eu vou criar um método, vamos lá. “func configuraSMS” e eu vou retornar “MFMessageComposeViewController”, é esse que eu vou retornar pra eu conseguir exibir esse componente na home do nosso app.

[02:40] Então vamos criá-lo, “let componenteMensagem” é do tipo “MFMessageComposeViewController”, que é o que nós precisamos retornar aqui. Vou instanciá-lo e já vou dar um “return” desse componente pra ele não apitar nenhum erro pra nós.

[03:02] Só que esse componente de SMS, assim como a câmera, que nós já implementamos, nós não conseguimos testar esse componente utilizando o simulador do Xcode, pra testar nós precisamos usar um device de verdade, um iPhone mesmo. Então eu preciso fazer uma tratativa aqui, porque se o desenvolvedor tentar abrir esse componente aqui no simulador, vai dar crash. Então é legal nós deixarmos com uma verificação.

[03:28] Como eu faço essa verificação? Através de um if mesmo, então vou colocar um “if MessageComposeViewController”, se ele pode enviar algum texto, “canSendText”, esse aqui.

[03:40] Se for possível, ou seja, se ele estiver utilizando um iPhone mesmo, nós fazemos toda essa implementação que nós programamos aqui. E se não for possível? O que nós vamos fazer? Se não for possível eu vou dar um “return” aqui, “nil”. Não é possível.

[03:59] Só que pra eu dar esse “return nil” eu não posso deixar o retorno do método desse jeito, porque aqui nós estamos falando, “vou retornar esse componente de mensagem”. Pra conseguirmos dar esse nil, eu tenho que deixar o retorno aqui como optional, então eu vou colocar aqui o ponto de interrogação, porque eu não sei, eu vou tentar retornar, mas não tenho certeza, então vou deixar como optional.

[04:23] Só que a consequência disso é que na hora que nós formos implementar, nós vamos ter que tratar pra ver se ele realmente tem a instância desse componente ou não.

[04:32] Com o componente criado, o que vamos ter que fazer agora? Nós já podemos configurar algumas coisas nele, como, por exemplo, quando eu abrir o componente, eu quero que ele seja aberto com o número do celular do aluno já, para eu só digitar o texto e enviar. Ou eu também posso abrir com o número já configurado e com o corpo do texto configurado, tem várias configurações aí. A que vamos utilizar é pra abrir o componente de mensagem já com o número do aluno setado.

[05:00] Como que nós fazemos isso? Eu posso pegar esse “componenteMensagem.” essa propriedade aqui espera um array de string, que na verdade é o número dos alunos que eu vou passar nela, então vou dar um enter igual a um array e aqui eu passo o array de string.

[05:19] Então, nós precisamos do telefone do aluno pra setarmos nessa propriedade, pra quando eu abrir o componente de mensagem já apareça o número do aluno setado pra nós.

[05:30] Nós vamos passá-lo como parâmetro, vou colocar aqui “aluno”, é do tipo “Aluno” mesmo, só que como a propriedade telefone é optional, eu vou precisar tratá-la, então vou criar aqui um “guard let numeroDoAluno” é igual ao “aluno” que ele vai receber por parâmetro “.telefone”, repara que aqui ele é mesmo uma string optional, então nós precisamos tratar.

[05:56] Se eu conseguir, beleza, se não, ou seja, “else”, o que eu vou fazer? Eu vou dar um “return” do componente de mensagem, é isso que eu vou fazer. Agora que eu tenho já acesso ao número do aluno com segurança, eu vou passá-lo aqui, “numeroDoAluno”.

[06:17] Tem mais uma configuração que nós precisamos fazer ainda, nós temos aqui um método de delegate, mas ninguém está o implementando, nós não setamos o delegate ainda. Pra setar nós vamos pegar o “componenteMensagem.messageComposeDelegate”, essa propriedade aqui, e eu passo como “self”. É essa classe que vai implementar o método de delegate.

[06:45] Nós já fizemos a configuração que nós precisávamos nessa classe, agora é hora de implementarmos na home. Vamos voltar pra lá? Vou voltar pra cá. No case aqui de SMS é que nós vamos implementar. Então vou tirar esse “print”, que nós só utilizamos para testar, e agora nós vamos realmente implementar.

[07:04] Lembra, , que quando nós criamos, nós acabamos de criar esse método, nós deixamos o retorno dele como optional, está um ponto de interrogação aqui, ou seja, nós não sabemos se realmente vai ser retornado o componente, então agora nós precisamos tratar isso. Como nós fazemos? Através de um “if let” mesmo, então “if let componenteMensagem”. Agora nós precisamos ter acesso à essa classe “Mensagem” aqui.

[07:34] Pra ficar mais fácil eu vou criar uma variável aqui em cima mesmo, “mensagem” que é do tipo “Mensagem()”, já vou instanciá-la, e eu vou utilizá-la aqui embaixo, onde estamos configurando o nosso case. Aqui eu passo “mensagem”, que nós acabamos de criar, “.configuraSMS”. Agora eu preciso do aluno pra passar como parâmetro, só que nós ainda não temos acesso ao aluno, então nós precisamos pegá-lo agora de alguma forma.

[08:06] Aqui embaixo eu vou criar um “guard let” do “alunoSelecionado”. Como nós fazemos pra ter acesso à lista de alunos mesmo? Nós precisamos do “gerenciadorDeResultados.fetchedObjects”, ele vai retornar para mim uma lista de

alunos. Só que o que eu quero, não é uma lista de alunos, eu quero o aluno que eu fizer o long press, quero capturar esse aluno. Como nós fazemos pra pegá-lo?

[08:36] Eu preciso passar o gesture, que é o long press, através da tag dele, então eu consigo resgatar esse aluno. Olha o que eu vou fazer, eu vou pegar o "longPress.view.tag", assim eu consigo pegar o aluno que eu fiz o long press. Se não der certo eu vou dar um "else" e vou dar um "return" sem passar nada, então "return", não consegui pegar o aluno. Assim nós conseguimos ter acesso ao aluno.

[09:13] Agora que eu tenho acesso ao aluno através do gesture que nós fizemos, eu vou passá-lo aqui, então "alunoSelecionado". Primeira parte ok. Então se eu conseguir retornar o componente de mensagem, nós seguimos com a implementação que nós precisamos fazer.

[09:31] Como nós vamos fazer agora a implementação? Eu vou pegar o componente de mensagem que nós acabamos de criar e quem vai implementar o método dele é essa classe "Mensagem", que é o que nós configuramos, não é ela que tem esse método aqui? Então eu vou falar que o "messageComposeDelegate" é igual a classe "mensagem" que nós estamos utilizando aqui, então "mensagem".

[10:00] O próximo passo, é exibirmos esse componente na tela. Como nós vamos fazer isso? Através do present mesmo, então "self.present", eu vou passar a classe "componenteMensagem", animado "true", e no bloco de completion não vou fazer nada. Vamos ver o que ele está reclamando aqui. Como eu estou dentro de uma closure, eu vou precisar do self, então "self.mensagem". E aqui também, "self.mensagem".

[10:35] Como nós vimos, se eu rodar isso no simulador ainda não vai acontecer nada, porque nós não temos acesso. Só para testarmos vamos fazer o seguinte, vamos rodar primeiro no simulador e depois eu vou rodar no iPhone pra nós testarmos a implementação. Então primeiro no simulador, vou gerar aqui um build.

[10:53] Com o simulador aberto nós já vamos poder fazer esse teste e nós vamos ver que ele não vai entrar no "if". Nós podemos até debugar isso daqui, se eu colocar um breakpoint, eu vou fazer um long press aqui em cima de um aluno e vou clicar em "enviar SMS". Cliquei, ele vai disparar nesse cara que nós criamos.

[11:15] Ele está com uma linha verde aqui, que é o debug que nós colocamos. Se eu passar para próxima linha através desse botão, ele não vai entrar no "if", então ele já vai cair no "return nil", ou seja, não dá pra exibirmos esse componente no simulador. Então vou tirar o breakpoint.

[11:34] Como não dá pra exibirmos esse componente no simulador, eu vou rodá-lo no meu iPhone, eu vou escolher o meu iPhone e eu vou gerar um build. Vamos ver. Eu estou com o app rodando no meu iPhone mesmo e agora nós vamos testar. Vou fazer um long press aqui em cima do primeiro aluno e vou clicar em "SMS".

[11:57] E repara que ele abriu o componente de mensagem do iOS mesmo com o número de telefone já preenchido, que é a configuração que nós fizemos na classe "Mensagem". Então aqui nós configuramos quais eram os números que nós queríamos que aparecessem. Com isso nós conseguimos implementar o componente de mensagem. A seguir nós vamos continuar implementando outros componentes.