

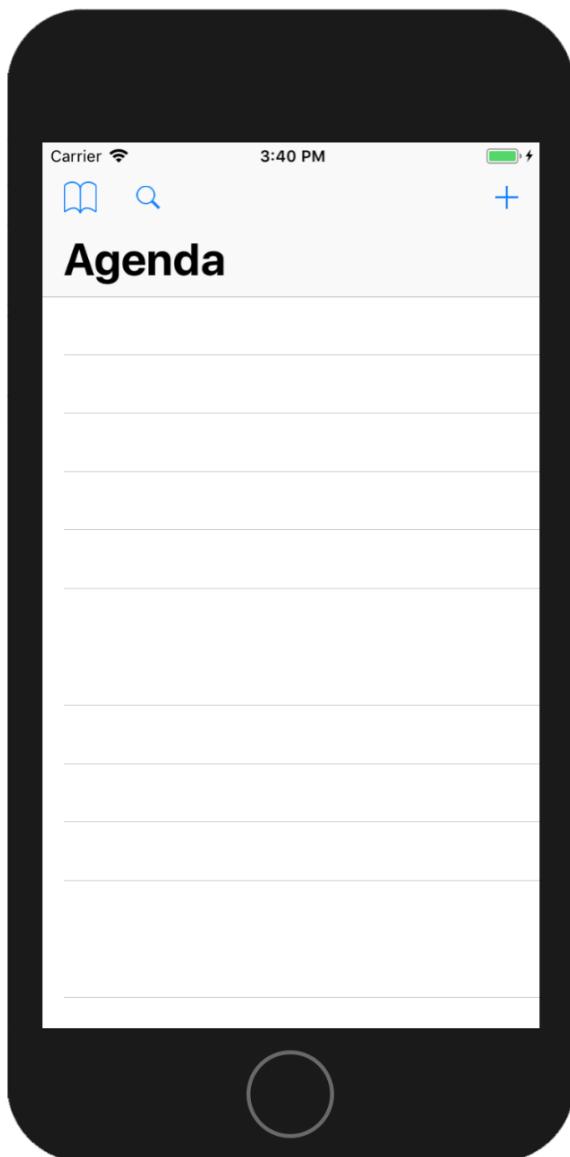
03

Acessando a câmera

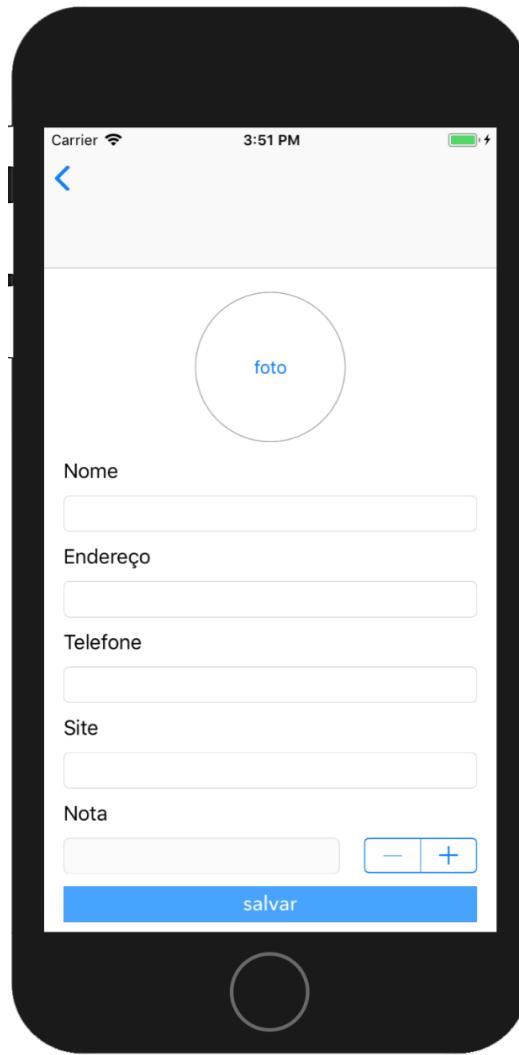
Para **começar**, vamos fazer o **download** da parte visual do projeto:

<https://github.com/alura-cursos/ios-recursos-nativos-parte-1-inicio/archive/master.zip> (<https://github.com/alura-cursos/ios-recursos-nativos-parte-1-inicio/archive/master.zip>)

Após concluir o download, vamos abrir e rodar o projeto:



A primeira feature que vamos implementar é o cadastro de alunos. Vamos clicar no botão "+" para conferir o formulário:



Aqui temos um formulário, onde vamos cadastrar as informações dos alunos. Repare que temos alguns componentes já conhecidos como: labels, textfields, buttons.

A primeira implementação que faremos será a utilização da câmera através do clique do botão "foto".

Para ter acesso as funcionalidades da câmera ou da biblioteca de fotos, utilizaremos a classe: **UIImagePickerController**.

```
@IBAction func buttonFoto(_ sender: UIButton) {
    let multimidia = UIImagePickerController()
    multimidia.sourceType = .camera
    self.present(multimidia, animated: true, completion: nil)
}
```

Repare que dessa forma que implementamos, estamos pedindo para o viewController apresentar o gerenciadorDeMultimidia

Vamos rodar o aplicativo para testar:

```

22  GraphicsServices
23  UIKit
159
24  Agenda
25  libdyld.dylib
26  ???
)
libc++abi.dylib: terminating with uncaught exception of type NSException
(lldb)

```

All Output ◊ Filter

Ocorreu uma exception ao clicar no botão **foto**.

Isso ocorreu porque o simulador do iOS não possui todas as funcionalidade que um iPhone/iPad de verdade tem, como por exemplo a câmera.

Precisamos fazer uma verificação para saber se o iOS consegue abrir a câmera ou não:

```

@IBAction func buttonFoto(_ sender: UIButton) {
    if UIImagePickerController.isSourceTypeAvailable(.camera) {
        let multimidia = UIImagePickerController()
        multimidia.sourceType = .camera
        self.present(multimidia, animated: true, completion: nil)
    }
}

```

Agora não ocorrerá erro ao clicar no botão **foto**.

Como acabamos de aprender, não é possível utilizar esse recurso no simulador. Portanto, precisaremos testar em um device de verdade.

Vamos testar:

```

libsystem_kernel.dylib:_abort_with_payload:
1 0x184f3c0808 <+0>; mov x16, #0x209
2 0x184f3c0808 <+0>; svc #0x80
3 0x184f3c0804 <+4>; svc #0x80
4 -> 0x184f3c080c <+8>; b.ln #0x184f3c08c0 ; <+32>
5 0x184f3c080c <+12>; stp x29, x30, [sp, #-0x10]!
6 0x184f3c0808 <+16>; mov x29, sp
7 0x184f3c0804 <+20>; bl 0x184f1e238 ; cerror_nocancel
8 0x184f3c0804 <+24>; mov sp, x29
9 0x184f3c080c <+28>; ldp x29, x30, [sp], #0x10
10 0x184f3c0808 <+32>; ret
11

```

Thread 9: signal SIGABRT

```

HangTracerDuration: 500
2017-12-01 10:31:47.476804 +0200 Agenda[1539:557834] refreshPreferences:
Action: ActivationLoggingTaskedOffByDA[0]
2017-12-01 10:31:50.306277 -0200 Agenda[1539:557834] [MC] Reading from public
effective user settings...
2017-12-01 10:31:59.455711 -0200 Agenda[1539:557239] [access] This app has crashed
because it attempted to access privacy-sensitive data without a usage description.
The Info.plist must contain an NSCameraUsageDescription key with a string
value explaining to the user how the app uses this data.
(lldb)

```

Repare que o app deu crash novamente. Isso ocorre por questões de segurança.

O iOS exige que o usuário autorize ou não a utilização desse recurso no nosso aplicativo.

Agora é a sua vez:

- Implemente a permissão para acessar a câmera no arquivo **Info.plist**