Aprendendo a programar para iOS

UIImagePickerController

An object that manages customizable, system-supplied user interfaces for taking pictures and movies on supported devices, and for choosing saved images and movies for use in your app.

Utilizando UIImagePickerController

- Gerencia interações do usuário e entrega o resultado destas interações para um “delegate”.
- O atributo “sourceType” diz de onde vem a imagem:

```
UIImagePickerControllerSourceTypeCamera
UIImagePickerControllerSourceTypePhotoLibrary
UIImagePickerControllerSourceTypeSavedPhotosAlbum
```

Utilizando UIImagePickerController

- Para que o sistema permita que nós utilizemos a câmera ou as imagens do dispositivo, devemos configurar o aviso de permissão para isto, no arquivo info.plist.
  - Privacy - Photo Library Usage Description
  - Privacy - Microphone Usage Description
  - Privacy - Camera Usage Description

Utilizando UIImagePickerController

1. Verificar se o dispositivo pode escolher fotos da fonte escolhida, chamando o método:
   `[UIImagePickerController isSourceTypeAvailable:TIPO];`
2. Verifique quais tipos de mídia estão disponíveis para a fonte de imagens que você vai usar chamando o método:
   `[UIImagePickerController availableMediaTypesForSourceType:TIPO]`
3. Configure a interface pra exibir a interface pra os típos de mídia que você quiser atribuindo um valor para o atributo “mediaTypes”, que é um array.
4. Apresente a interface, não esquecendo de configurar o delegate e os métodos para tratar o retorno.

Utilizando UIImagePickerController

- Apresentando interface:

```
UIImagePickerController *cameraUI = [[UIImagePickerController alloc] init];
[self presentViewController: cameraUI animated: YES completion:nil];
```

Métodos do delegate pra implementar:

```
(void)imagePickerControllerDidCancel:(UIImagePickerController *)picker;
- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary<NSString *,id> *)info;
```

Utilizando UIImagePickerController

```
- (BOOL) startCameraControllerFromViewController: (UIViewController*) controller
                                   usingDelegate: (id <UIImagePickerControllerDelegate,
UINavigationControllerDelegate>)delegate {

    if (([UIImagePickerController isSourceTypeAvailable:
          UIImagePickerControllerSourceTypePhotoLibrary] == NO)
        || (delegate == nil)
        || (controller == nil))
        return NO;
    UIImagePickerController *cameraUI = [[UIImagePickerController alloc] init];
    cameraUI.sourceType = UIImagePickerControllerSourceTypePhotoLibrary;

    cameraUI.mediaTypes =
    [UIImagePickerController availableMediaTypesForSourceType:
     UIImagePickerControllerSourceTypePhotoLibrary];
    cameraUI.allowsEditing = NO;

    cameraUI.delegate = delegate;
    [controller presentViewController: cameraUI animated: YES completion:nil];
    return YES;
}
```

Utilizando UIImagePickerController

```
- (void)imagePickerController:(UIImagePickerController *)picker didFinishPickingMediaWithInfo:(NSDictionary<NSString *,id> *)info {
    NSString *mediaType = [info objectForKey: UIImagePickerControllerMediaType];
    UIImage *originalImage, *editedImage, *imageToSave;

    // Handle a still image capture
    if ([mediaType isEqualToString:@"public.image"]) {
        editedImage = (UIImage *) [info objectForKey:
                                   UIImagePickerControllerEditedImage];
        originalImage = (UIImage *) [info objectForKey:
                                     UIImagePickerControllerOriginalImage];

        if (editedImage) {
            imageToSave = editedImage;
        } else {
            imageToSave = originalImage;
        }

        UIImageWriteToSavedPhotosAlbum (imageToSave, nil, nil , nil);
    }

    // Handle a movie capture
    if ([mediaType isEqualToString:@"public.movie"]) {

        NSString *moviePath = [[info objectForKey: UIImagePickerControllerMediaURL] path];

        if (UIVideoAtPathIsCompatibleWithSavedPhotosAlbum(moviePath)) {
            UISaveVideoAtPathToSavedPhotosAlbum(moviePath, nil, nil, nil);
        }
    }

    [[picker parentViewController] dismissViewControllerAnimated:YES completion:nil];
}
```

Utilizando AFNetworking – POST – Multipart Request

```
AFHTTPRequestOperationManager *manager = [AFHTTPRequestOperationManager manager];
NSDictionary *parameters = @{@"foo": @"bar"};
NSURL *filePath = [NSURL fileURLWithPath:@"file://path/to/image.png"];
[manager                    POST:@"http://example.com/resources.json"
                 parameters:parameters
constructingBodyWithBlock:^(id<AFMultipartFormData> formData) {
     [formData appendPartWithFileURL:filePath name:@"image" error:nil];
}                    success:^(AFHTTPRequestOperation *operation, id responseObject) {
     NSLog(@"Success: %@", responseObject);
}                    failure:^(AFHTTPRequestOperation *operation, NSError *error) {
     NSLog(@"Error: %@", error);
}];
```

- Arquivos podem ser adicionadas via NSData ou NSURL:
  - `appendPartWithFileURL:name:error:`
  - `appendPartWithFileData:name:fileName:mimeType:`

Conteúdo da Aula