

Aprendendo a programar para iOS



UIAlertController

An object that displays an alert message to the user.

Utilizando UIAlertController

- Classes utilizadas pra exibir mensagens de alerta para o usuário.
- Você cria o alerta escolhendo um estilo, cria ações (botões) escolhendo o estilo e métodos de callback, adiciona as ações ao alerta e exibe o mesmo.

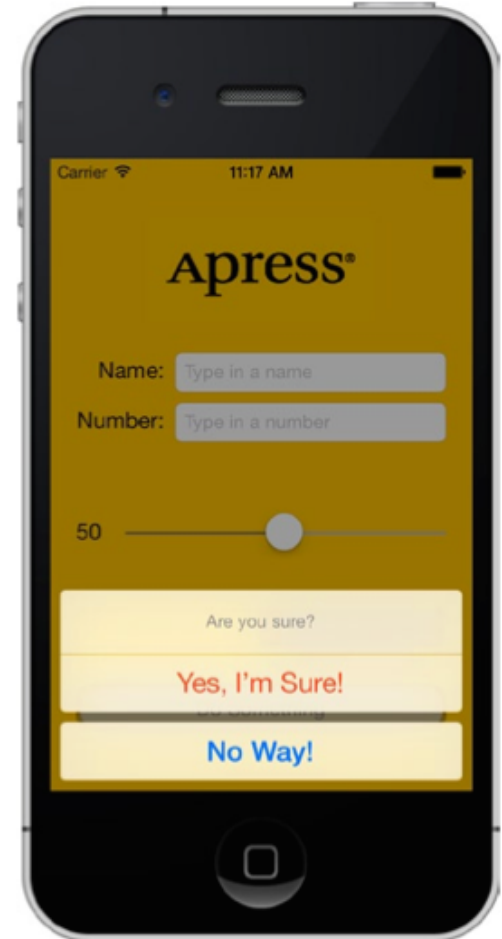
Utilizando UIAlertController

- Estilo do UIAlertController são:
 - UIAlertControllerStyleAlert
 - Alerta no centro da tela
 - UIAlertControllerStyleActionSheet
 - Alerta na parte inferior da tela

Utilizando UIAlertController

- Estilo do UIAlertAction são:
 - `UIAlertActionStyleDefault`
 - Botão padrão.
 - `UIAlertActionStyleCancel`
 - Estilo que indica cancelamento da operação sem mudar nada.
 - `UIAlertActionStyleDestructive`
 - Estilo que indica que esta ação pode perder ou deletar dados.

Utilizando UIAlertController



Utilizando UIAlertController

- Criando uma UIAlertController com um botão

```
UIAlertController *alertController = [UIAlertController
    alertControllerWithTitle:@"Opa"
                        message:@"Você deseja realmente fazer um alerta? "
    preferredStyle:UIAlertControllerStyleAlert];

UIAlertAction* okButton = [UIAlertAction actionWithTitle:@"Não"
                        style:UIAlertActionStyleDefault
                        handler:nil];

[alertController addAction:okButton];

[self presentViewController:alertController animated:YES completion:nil];
```

Utilizando UIAlertController

- Adicionando ações no clique dos botões.

```
UIAlertAction* cancelButton2 = [UIAlertAction  
    initWithTitle:@"Sim"  
    style:UIAlertActionStyleDefault  
    handler:^(UIAlertAction *action){  
        [SVProgressHUD show];  
    }];
```


UITableView

A view that presents data using rows arranged in a single column.

Utilizando UITableView

- Classe utilizada para exibir uma lista de itens em uma única coluna.
- É uma subclasse de UIScrollView, o que permite que o usuário faça “scroll” na lista verticalmente.
- Toda célula da lista é exibida utilizando UITableViewCell, que possui por padrão um título, imagem e uma view adicional no canto direito. Utilizando UITableView

Utilizando UITableView

- As UITableViewCell podem ser customizadas no storyboard ou utilizando Subclasses e arquivos XIB para a interface.
- UITableViews podem entrar em modo de edição, permitindo que usuários removam/ordenem/adicionem itens.
- Uma tabela é feita com uma ou mais “Sections”, cada “Section” contendo uma ou mais “Células”.

Utilizando UITableView

- Métodos avançados para sobrecarga:

```
(NSInteger)numberOfSectionsInTableView:(UITableView *)tableView;
```

```
(NSString *)tableView:(UITableView *)tableView  
titleForHeaderInSection:(NSInteger)section;
```

```
(UIView *)tableView:(UITableView *)tableView  
viewForHeaderInSection:(NSInteger)section;
```

```
(CGFloat)tableView:(UITableView *)tableView  
heightForHeaderInSection:(NSInteger)section;
```

```
(void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:  
(NSIndexPath *)indexPath;
```

Utilizando UITableView

- Métodos utilitários:

```
[self.tableView setEditing:YES animated:YES];
```

```
[self.tableView reloadData];
```

```
[self.tableView reloadSectionIndexTitles];
```

```
[self.tableView deselectRowAtIndexPath:indexPath animated:YES];
```

Material

- Material desta aula: bit.ly/aula-ios-4

Exercícios

- Crie uma aplicação com uma table view com Sections para regiões do Recife e células para os bairros.
- Valores a serem exibidos devem ser armazenados em uma propriedade.
- Ao selecionar uma célula, um UIAlertView deve ser exibido, perguntando se aquela célula deve ser deletada, ou se a tela de detalhes deve ser exibida.
- Se o usuário escolher deletar, a tableView deve ser recarregada sem a célula deletada.
- Se o usuário escolher exibir detalhes, uma nova viewController deve ser exibida utilizando UINavigationController
- <http://bit.ly/ex1-aula-4>

AFNetworking

AFNetworking is a delightful networking library for iOS and Mac OS X. It's built on top of the Foundation URL Loading System, extending the powerful high-level networking abstractions built into Cocoa.

Utilizando AFNetworking

- Biblioteca AFNetworking
 - Instalação via Cocoapods:
 - Podfile: `pod "AFNetworking", "~> 2.0"`
 - Instalação via Arquivos
 - Baixar ZIP do github: github.com/AFNetworking/AFNetworking
 - Inclua arquivos da pasta AFNetworking no projeto
- Utilização
 - `#import` nos arquivos `.h` necessários
 - Os métodos de rede são assíncronos, e utilizam blocos para funções de callback.
 - Um bloco é uma chamada de método assíncrona.
 - Pode retornar um resultado qualquer ou um erro.
 - A idéia por trás do bloco é permitir que enquanto uma requisição de rede seja realizada, a aplicação não pare o seu funcionamento

Utilizando AFNetworking

- Comunicação REST, padrão em APIs, utiliza chamadas HTTP comuns, que podem ser:
 - GET
 - POST
 - PUT
 - DELETE
- Para enviarmos arquivos, precisamos utilizar uma requisição HTTP Multipart

Utilizando AFNetworking - GET

```
AFHTTPRequestOperationManager *manager = [AFHTTPRequestOperationManager
manager];
[manager GET:@"https://teste-aula-ios.herokuapp.com/comments.json"
parameters:nil
success:^(AFHTTPRequestOperation *operation, id responseObject) {
    NSLog(@"JSON: %@", responseObject);
} failure:^(AFHTTPRequestOperation *operation, NSError *error) {
    NSLog(@"Error: %@", error);
}];
```

- responseObject é uma string, se o servidor retornar texto ou HTML, e é um Hash ou Array, se o servidor retornar JSON

Utilizando AFNetworking

- Servidor no endereço:
 - <https://teste-aula-ios.herokuapp.com/comments.json>
- Novo projeto, chamem, no view did appear, um GET no endereço:
 - <https://teste-aula-ios.herokuapp.com/comments.json>
- Caso o endereço não fosse HTTPS, estaríamos com problema.

Utilizando AFNetworking

- Para acessar endereços sem HTTPS

▼ App Transport Security Settings	▲▼	Dictionary	(1 item)
▼ Exception Domains	▲▼	Dictionary	(4 items)
▼ pos-mobile-ios.herokuapp.com		Dictionary	(3 items)
NSIncludesSubdomains		Boolean	YES
NSTemporaryExceptionAllowsInsecureHTTPLoads		Boolean	YES
NSTemporaryExceptionMinimumTLSVersion		String	TLSv1.1

Utilizando AFNetworking

- Exercício:
 - No viewDidLoadAppear, mostrar HUD, e pegar JSON no endereço:
 - <https://teste-aula-ios.herokuapp.com/comments.json>
 - Ao pegar itens, mostrar em table-view o nome e descrição.

Utilizando AFNetworking

- Imagens:
 - No AFNetworking, existe uma Category em UIImageView que carrega imagens da rede, pela URL.
 - Adicione arquivos em UIKit+AFNetworking
 - `#import "UIImageView+AFNetworking.h"`
 - UIImageView ganha métodos:
 - setImageWithURL:
 - setImageWithURL: placeholderImage:

Utilizando AFNetworking

- Exercício:
 - Adicionar tableViewCell customizada com UIImageView, e mostrar as imagens com placeholder, nome, comentário e hora

Utilizando AFNetworking - POST

```
AFHTTPRequestOperationManager *manager = [AFHTTPRequestOperationManager manager];
NSDictionary *parameters = @{@"foo":@"bar"};
[manager POST:@"http://example.com/resources.json"
 parameters:parameters
 success:^(AFHTTPRequestOperation *operation, id responseObject) {
    NSLog(@"JSON: %@", responseObject);
} failure:^(AFHTTPRequestOperation *operation, NSError *error) {
    NSLog(@"Error: %@", error);
}];
```

- Parameters vai ser mandado como parâmetros HTML normais.

Utilizando AFNetworking - POST

```
AFHTTPRequestOperationManager *manager = [AFHTTPRequestOperationManager manager];
manager.requestSerializer = [AFJSONRequestSerializer serializer];
NSDictionary *parameters = @{@"foo":@"bar"};
[manager POST:@"http://example.com/resources.json"
 parameters:parameters
 success:^(AFHTTPRequestOperation *operation, id responseObject) {
    NSLog(@"JSON: %@", responseObject);
} failure:^(AFHTTPRequestOperation *operation, NSError *error) {
    NSLog(@"Error: %@", error);
}];
```

- Parameters vai ser mandado como JSON

Utilizando AFNetworking – POST – Multipart Request

```
AFHTTPRequestOperationManager *manager = [AFHTTPRequestOperationManager manager];
NSDictionary *parameters = @{@"foo":@"bar"};
NSURL *filePath = [NSURL URLWithString:@"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:`

Exercício

- Fazer leitura e postagem de comentários.
- Ao exibir 20ª comentário, deve-se solicitar a próxima página.