UNIVERSITY everis

iOS

Tópicos Avançados

Swift

Introdução a plataforma

iOS

UNIVERSITY everis

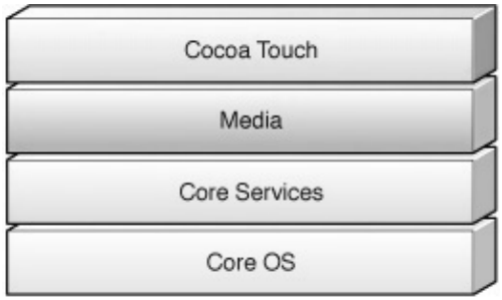
01

Arquitetura da Plataforma

UNIVERSITY everis

Arquitetura da plataforma

A arquitetura do iOS e formada pelas camadas: ´ Core OS, Core Services, Media e Cocoa Touch



- Cocoa Touch: Principais frameworks, tais como multitarefa, serviço de notificação Apple Push e diversos serviços de alto nível do sistema.
- A camada Media contem as tecnologias de gráfico, áudio e vídeo. Nesta camada o desenvolvedor pode utilizar o framework UIKit (User Interface Kit) que oferece varias tecnologias de gráficos e animações.
- A camada Core Services contem os serviços fundamentais do sistema que todos os aplicativos utilizam. As principais tecnologias disponíveis na camada Core Services são: grand central dispatch, in-app purchase, SQLite e XML support.
- A camada Core OS contem características de baixo nível, como Threading, Networking, Acesso ao sistema de arquivos, Alocação de memória, Cálculos matemáticos.

Boas praticas

Sinalizar dificuldade para o time, assim obterá sucesso, no projeto e na carreira.

o Voltar ou Fechar

Botão voltar no NavigationController, você pode usar o X no canto esquerdo pra fechar a tela, se não for usar no NavigationController o voltar

o Obtendo mais registros

No final da lista, pode-se incluir um botão "ver mais...", não é uma boa prática usar scrool infinito, pode realizar infinitas chamadas desnecessarios só arrastando a tela pra baixo.

o Carregando imagens pra um UIImagemView

Cuidado com distorções de imagens de pessoas e objetos, uma solução seria reduzir/ampliar mantendo proporções do original.

o Modularizando por feature

Podemos replicar nossa arquitetura de pastas de forma modular, assim ela pode ser reaproveitável, caminhando para se tornar independente, testavel e talvez um POD independente no futuro,

Ex.: Módulo de login, módulo de filmes, modulo de chamadas de API

o Abstrações quando possível, usar, em MVC é comum usar DAO para isto.

UNIVERSITY everis

Boas praticas

Extensions: Falta de extension Não seria uma boa prática:

o class ViewController: UIViewController, UICollectionViewDataSource, UICollectionViewDelegateFlowLayout, UISearchBarDelegate, UICollectionViewDelegate {

…

}

o Usando extension no mesmo arquivo:

extension ViewController: UICollectionViewDataSource {

...

}

UNIVERSITY everis

Boas praticas

Extensions: Usando extension em uma pasta communs/extensions

```
//UIImageView+extension
 extension UIImageView {
     func load(url: URL) {
       ...
     }
   }
//Int+extension
   extension Int {
     func toString() -> String {
       return "\(self)"
    }
   }
```

UNIVERSITY everis

Boas praticas

Qualidade e Performance, gerenciamento de memória

- https://dev.to/xreee/gestao-de-memoria-em-ios-e-descobertas-de-leaks-com-instruments-2eg4
- **Memory Leak**, analisando com **Memory Graph** e **Instruments**
- **Memory Leak**, aplicando **weak** e **deinit** para liberar objetos da memória

- **SwiftLint será abordado em outro momento**, mas ele aponta weak a ser acrescentado no código, como uma das várias regras para um código com qualidade, que aplica boas praticas.

UNIVERSITY everis

Curiosidades

UIkit vs SwiftUI

- **UIkit** exige mais domínio do código, enquanto **SwiftUI** é mais fácil pra fazer tela, porém exige maior organização e interpretação dos códigos aninhados.

- https://stackshare.io/stackups/swiftui-vs-uikit

- https://www.raywenderlich.com/4919757-your-first-ios-and-swiftui-app/lessons/2

UNIVERSITY everis

Curiosidades

Objective-C 2.0

o Livro: "Programming in Objective-C 2.0”

o Um exemplo de similaridade com swift

```
- (void)someMethod {
  NSString *myString = @"An interesting string";
  self.someString = myString; //Objective-C 2.0, as vezes pode-se confundir class com property
  // or
  [self setSomeString:myString]; //Objective-C raiz, neste caso, usa-se mais assim, apesar da facilidade acima
}
```

Curiosidades

RxSwift

- **É Uma forma de observar um objeto e ser reativo a qualquer ação nele, atualmente o Swift já faz isto e nos fornece o retorno via delegates, porém o RxSwift centraliza este retorno, assim teremos menos delegates, porém um código mais verboso. https://www.thedroidsonroids.com/blog/rxswift-by-examples-1-the-basics**

```
import RxCocoa
import RxSwift
searchBar
    .rx.text // Observable property thanks to RxCocoa
    .orEmpty // Make it non-optional
    .subscribe(onNext: { [unowned self] query in // Here we will be notified of every new value
        self.shownCities = self.allCities.filter { $0.hasPrefix(query) } // We now do our "API Request" to find cities.
        self.tableView.reloadData() // And reload table view data.
    })
    .addDisposableTo(disposeBag)
```

Curiosidades

Persistindo dados

- **CoreData -> SQLite vs Realm -> Json**

Normalmente o CoreData está linkado com o SQLite; este é um bando de dados relacional, enquanto que o Realm está linkado com BD no formato Json, normalmente não relacional.

- **https://www.raywenderlich.com/7569-getting-started-with-core-data-tutorial**
- **https://agilie.com/en/blog/coredata-vs-realm-what-to-choose-as-a-database-for-ios-apps**

UNIVERSITY everis

Arquitetura do projeto

MVC e MVVM

- **MVC é mais antiga, pouca usada. MVVM, relativamente moderna, propondo maior testabilidade e organização do código, vejamos algumas diferenças.**

- **https://www.guru99.com/mvc-vs-mvvm.html (tabela de comparação)**

- **https://www.bacancytechnology.com/blog/mvc-vs-mvp-vs-mvvm**

Arquitetura do projeto

MVC	MVVM
Controller é o ponto de entrada para o aplicativo.	A view é o ponto de entrada para o aplicativo.
Um para muitos relacionamentos entre o Controller & View.	Um a muitos relacionamentos entre o View & View Model.
View Não tem referência para o Controller	View tem referências ao View-Model.
MVC é modelo antigo	MVVM é um modelo relativamente novo.
Difícil de ler, alterar, testar a unidade e reutilizar este modelo	O processo de depuração será complicado quando tivermos ligações de dados complexas.
O componente MVC Model pode ser testado separadamente	Fácil para testes de unidade separados e código é orientado a eventos.

Arquitetura do projeto

CleanSwift

o CleanSwift é uma organização por cenários, centralizando a interação e ordenando um fluxo unidirecional VIP, é possível obter um suporte, utilizando um template integrado ao xCode, disponível no link abaixo

https://clean-swift.com/handbook/

o Curiosidades:

Tanto o MVVM, quanto CleanSwift pode ser usado com o Coordinator, para controlar a navegação, temos assim, por exemplo, o MVVM-C

https://benoitpasquier.com/coordinator-pattern-navigation-back-button-swift/

https://medium.com/sudo-by-icalia-labs/ios-architecture-mvvm-c-coordinators-3-6-3960ad9a6d85

UNIVERSITY everis

Atividade prática

Aplique no seu projeto de Filmes, a arquitetura MVVM ou Clean

. Atividade prática 01

o Aplique a arquitetura MVVM no seu projeto de Filmes, segue um exemplo de uso da arquitetura:

https://github.com/danielcolnaghi/TheMovie/tree/master/DCTheMovie

. Atividade prática 02

o Aplique a arquitetura CleanSwift no seu projeto de Filmes, segue um exemplo de uso da arquitetura:

https://github.com/Clean-Swift/CleanStore

UNIVERSITY everis

Consulting, Transformation, Technology and Operations