P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

2022

PDM II – Trabalho Prático

PROGRAMAÇÃO PARA DISPOSITIVOS MÓVEIS II

ANDRÉ PINTO (8200613) / SERGIO FÉLIX (8200615) / BRUNO JOSUÉ (8200210)

Índice

Introdução

No âmbito da disciplina de Programação para Dispositivos Mobile II do 2º ano, 1º semestre do Curso Técnico Superior Profissional em Desenvolvimento para a Web e Dispositivos Moveis, foi pedido o desenvolvimento de uma aplicação nativa em IOS, que se contempla todos os pontos pedidos no enunciado do trabalho.

Este trabalho teve como tema uma aplicação de notícias aeroespaciais, onde os utilizadores poderiam visualizar as notícias, adicionar aos favoritos, dar like e comentar as publicações, e alterar o tempo de retenção dos dados em cache.

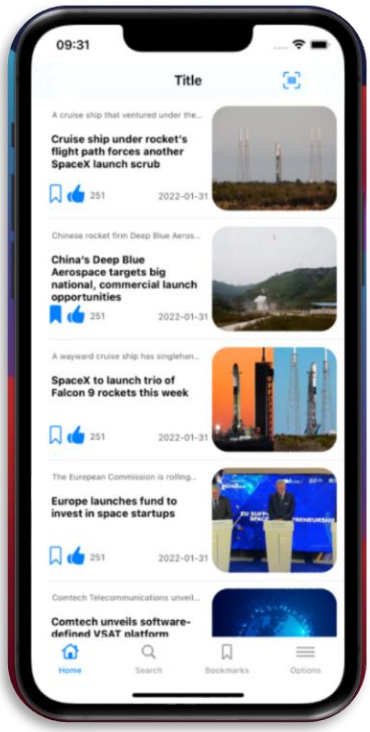
Para alem destas funcionalidades, a aplicação, teria de implementar mecanismos de persistência de dados, notificações, definições e todos os dados seriam obtidos através da API (SpaceFlightNewsApi).

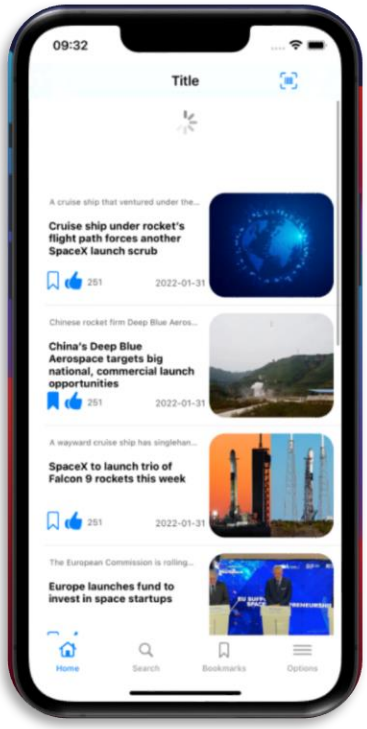
Implementação

Apresentação

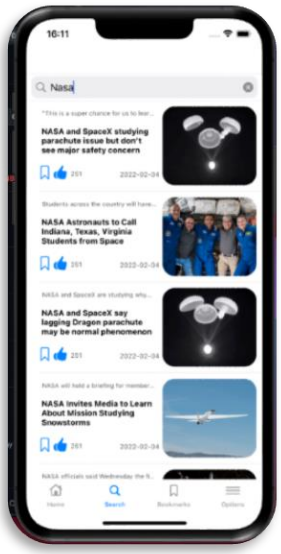
Tal como indicado anteriormente, esta aplicação, visa possibilitar ao utilizador a leitura de notícias aeroespaciais, provenientes da API SpaceFlightNews. Para tal, a aplicação desenvolvida conta com quatro áreas distintas, em que a sua navegação é através de uma "NavBar".

A primeira área, "HomePage" é responsável por lista as últimas notícias. Através de um "SwipeDown" é possível forçar a atualização dos dados. Para tal é efetuado um pedido á API, ela retorna as 10 noticias mais recentes e posteriormente são armazenadas no "CoreData". Caso seja uma nova noticia, ela será adicionada, caso contrário irá atualizar os dados em cache. A listagem das notícias é efetuada numa "TableView", com uma "ViewCell" personalizada baseada nos dados armazenados em cache no "CoreData", é possível adicionar a notícia aos favoritos e a imagem de capa é carregada de forma assíncrona. Para além destas funcionalidades, é possível navegar para o detalhe da notícia e abrir a camara para sacanear o QRCode de partilha.

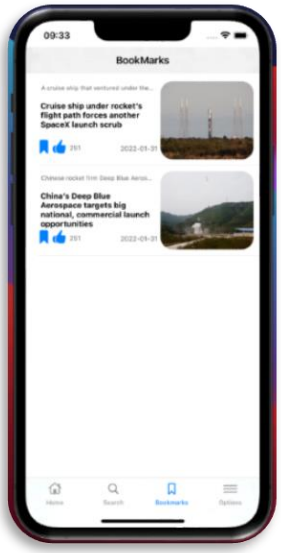




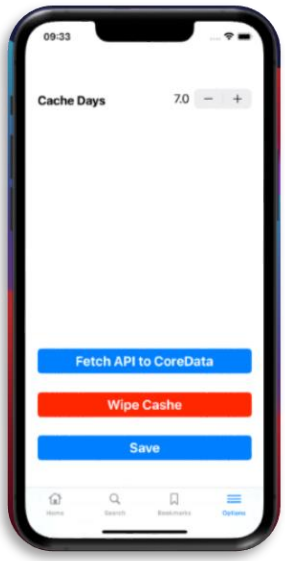
SearchPage – Página responsável para realizar uma pesquisa. Realiza um pedido à API especificando o parâmetro de pesquisa "articles?title_contains=VARIAVEL". Após retornar os dados, estes são armazenados em cache e listados numa "TableView", reutilizando a "ViewCell" anteriormente usada.



BookPage - Área responsável por listar as notícias adicionada aos favoritos. Tal como a página anterior, lista os resultados numa "TableView", reutilizando a "ViewCell" anteriormente usada.



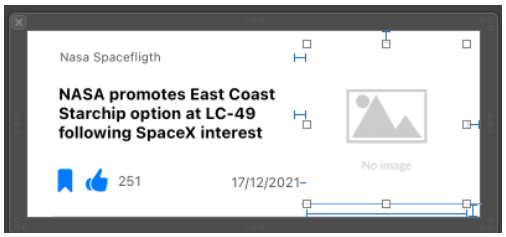
Options - Esta área é responsável por gerir as configurações da aplicação ("data de expiração dos dados"). Nela também é possível realizar algumas funcionalidades de manutenção, tais como, wipe da cache ou force na obtenção dos dados tal como no SwipeDown na HomePage.



Página de Detalhe – Responsável por mostrar todos os detalhes da notícia, como Imagem de capa, Título, Fonte, Sumário. Ainda é possível adicionar ou remover dos favoritos e realizar a partilha através do QRCode.

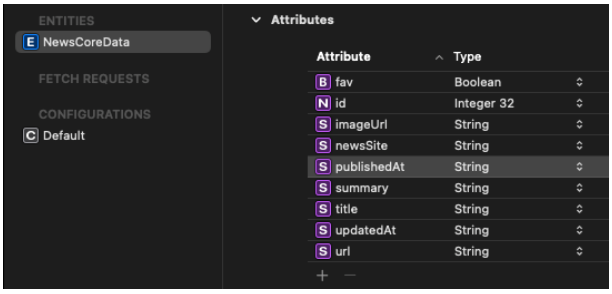
Layout Adaptativo

Tal como pedido no enunciado, esta aplicação contempla um suporte para ecrãs de diferentes dimensões. Para tal foram adicionadas "Constraints" de modo a fazer com que o layout se ajuste ao ecrã do telemóvel.



Core Data

Um dos objetivos principais propostos seria o armazenamento em cache dos dados obtidos da API, de modo e reduzir o consumo de internet. Para essa implementação utilizamos a ferramenta "CoreData" que nos possibilita o armazenamento dos dados de forma estruturada na aplicação para uma posterior utilização.



Os parâmetros da entidade criada, são os mesmo dos parâmetros recebidos da API, adicionando o campo favorito para guardar caso a notícia esteja como favorita.

Integração API

Para a integração com a API, foi criado uma classe, responsável por permitir a comunicação com a API abstraindo assim a sua implementação. Após realizar o pedido, os dados são convertidos para o modelo de dados criado "NewsModel".

```swift
import Foundation

final class APIService{
    static let shared = APIService()

    struct Constants {
        static let topHeadlinesURL = URL(string: "https://api.spaceflightnewsapi.net/
    }

    private init() {}

    public func getTopArticles(completion: @escaping (Result<[NewsModel], Error>)-> V
        guard let url = Constants.topHeadlinesURL else{
            return
        }

        let task = URLSession.shared.dataTask(with: url) { data, _, error in
            if let error = error {
                print("REQUEST ERROR ON START")
                completion(.failure(error))
            }
            else if let data = data {
                do {
                    let result = try JSONDecoder().decode([NewsModel].self, from: data
                    print("REQUEST: \(result.count)")
                    completion(.success(result))
                }
                catch{
                    print("REQUEST ERRORrrrrrrr")
                    completion(.failure(error))
                }
            }

        }
        task.resume()
    }
```

```swift
import UIKit

struct NewsModel: Codable {
    let id: Int32
    let title : String
    let url : String?
    let imageUrl : String
    let newsSite : String
    let summary : String
    let publishedAt: String?
    let updatedAt: String
    let fav: Bool?


}
```

Definições (APP Delegate)

A Aplicação também possibilita a funcionalidade de colocar definições. Como implementação, utilizamos a ferramenta nativa "UserDefault". Estas configuração são inicialmente carregadas, quando a aplicação é inicializada e salvas quando a aplicação é fechada. Para tal realizamos estes procedimentos no AppDelegate. Como pedido no enunciado apenas foi implementada a definição "Tempo de expiração" dos dados em cache. Facilmente poderá ser adicionado mais.

Operações Assíncronas

Neste ponto, utilizamos I tarefas assíncronas na aplicação. O carregamento de imagens de forma assíncrona. Após obter a imagem é enviado um uma tarefa para a main Queue para mostrar a imagem na "ImageView".

```swift
DispatchQueue.global(qos: .userInitiated).async {
    // Fetch Image Data
    if let data = try? Data(contentsOf: url) {
        DispatchQueue.main.async {
            // Create Image and Update Image View
            self.imgCover.image = UIImage(data: data)
        }
    }
}
```

Geração e Leitura QRCode

```
override func viewDidLoad() {
    super.viewDidLoad()
    lblTexto.text = String(newsID)

    if let QRCodeImage = createQRFromString(str: String(newsID)){
        let img = UIImage(ciImage: QRCodeImage, scale: 0, orientation: .down)
        self.imgQRCode.image = img
    }

}

func createQRFromString(str: String) -> CIImage?{
    let stringData = str.data(using: .utf8)

    let filter = CIFilter(name: "CIQRCodeGenerator")

    filter?.setValue(stringData, forKey: "inputMessage")

    filter?.setValue("H", forKey: "inputCorrectionLevel")

    return filter?.outputImage
}
```

Esta funcionalidade está presente na página de detalhe da notícia. Após clicar no botão de QRCode, irá abrir uma "View" que irá mostrar o id da notícia e o respetivo QRCode. Para gerar, foi utilizado ferramentas nativas no swift "CIFilter - CIQRCodeGenerator".

No que toca á leitura / utilização da camara, foram implementadas todas as funções necessárias para o seu bom funcionamento. No entanto, foram encontrados erro que não os conseguimos corrigir. Consequentemente não foi possível testar esta funcionalidade e esta não ficou terminada.

Notificações

Como previsto, as notificações foram implementadas na página "Home". A segunda função, é responsável por garantir as permissões de utilização, enquanto a primeira é responsável por contruir, agendar e adicionar a notificação à queue de notificações.

Infelizmente esta funcionalidade não ficou 100% terminada visto que por algum motivo desconhecido, a pesar das notificações serem adicionas á queue, as mesmas não são mostradas.

```
130     func scheduleNotification(){
131         let content = UNMutableNotificationContent()
132         content.title = "Title"
133         content.body = "Body!"
134
135         //Fire in 90 sec
136         let trigger = UNTimeIntervalNotificationTrigger(timeInterval: 5, repeats: false)
137
138         //create the request
139         let uuidString = UUID().uuidString
140         let request = UNNotificationRequest(identifier: uuidString, content: content, trigger: trigger)
141
142         //Schedule the request whith the system
143         let notificationCenter = UNUserNotificationCenter.current()
144         notificationCenter.add(request){ (error) in
145             if error != nil{
146                 print("error on notification")
147             }
148             print("notification add ok")
149
150         }
151     }
152
153     func handlePermissionForNotification(){
154         let notificationCenter = UNUserNotificationCenter.current()
155         notificationCenter.delegate = self
156         notificationCenter.requestAuthorization(options: [.alert, .sound, .badge, .provisional]) { (granted, error) in
157             if let error = error {
158                 print(error)
159             }
160             print("Tenho permissoes?: \(granted)")
161         }
162
163     }
```

Melhorias

Durante o tempo que tivemos para realizar este trabalho, não nos foi possível realizar todas as ideias previstas para o trabalho. Assim, transitam para possíveis melhorias do projeto, tais como:

- Expiração de dados
- Comentários
- Número de Likes
- Push Notifications

Conclusão

Começamos este trabalho com boas expectativas para a sua implementação. No entanto, tivemos várias dificuldades durante a seu desenvolvimento, o que nos levou a não conseguir cumprir todos os objetivos esperados.

Estas dificuldades vão desde o acesso aos materiais necessários, até ao facto de nunca ter anteriormente trabalhado com Swift. Consequentemente, houve a exigência de um esforço extra em reduzir a curva de aprendizagem e conseguir realizar as funcionalidades pretendidas em tempo útil.

Graças á dedicação dos elementos do grupo, foi possível implementar grande parte das funcionalidades pedidas, que do qual nos orgulhamos do resultado.

Bibliografia

- IOS Developers - https://developer.apple.com/
- Stackoverflow - https://pt.stackoverflow.com/
- Youtube - https://www.youtube.com/