Лабораторная работа №11 Swift UI

1. Создать приложение используя Swift UI
2. Приложение должно состоять из 2 экранов: список информации в зависимости от варианта (делать через таблицу), экран подробного просмотра (переход кликнув на ячейку таблицы)
3. Информация отображаемая в списке должна загружаться с файла json(вспомогательный листинг ниже)
4. В приложении нужно использовать 3 нестандартных View помимо TableView.

Вариант

1 Developer

2 Food

3 Fashion

4 Shop

5 Transport

6 Music

7 Film

8 Messenger

9 ComputerGame

10 Sport

11 Work

12 Hobbies

Пример листинга для выгрузки информации с json

|  |
| --- |
| import UIKit  import SwiftUI  import CoreLocation  let landmarkData: [Landmark] = load("landmarkData.json")  func load<T: Decodable>(\_ filename: String) -> T {  let data: Data    guard let file = Bundle.main.url(forResource: filename, withExtension: nil)  else {  fatalError("Couldn't find \(filename) in main bundle.")  }    do {  data = try Data(contentsOf: file)  } catch {  fatalError("Couldn't load \(filename) from main bundle:\n\(error)")  }    do {  let decoder = JSONDecoder()  return try decoder.decode(T.self, from: data)  } catch {  fatalError("Couldn't parse \(filename) as \(T.self):\n\(error)")  }  }  final class ImageStore {  typealias \_ImageDictionary = [String: CGImage]  fileprivate var images: \_ImageDictionary = [:]  fileprivate static var scale = 2    static var shared = ImageStore()    func image(name: String) -> Image {  let index = \_guaranteeImage(name: name)    return Image(images.values[index], scale: CGFloat(ImageStore.scale), label: Text(name))  }  static func loadImage(name: String) -> CGImage {  guard  let url = Bundle.main.url(forResource: name, withExtension: "jpg"),  let imageSource = CGImageSourceCreateWithURL(url as NSURL, nil),  let image = CGImageSourceCreateImageAtIndex(imageSource, 0, nil)  else {  fatalError("Couldn't load image \(name).jpg from main bundle.")  }  return image  }    fileprivate func \_guaranteeImage(name: String) -> \_ImageDictionary.Index {  if let index = images.index(forKey: name) { return index }    images[name] = ImageStore.loadImage(name: name)  return images.index(forKey: name)!  }  } |