# Лекция 14
# Мобильные приложения

Разработка интернет приложений

Канев Антон Игоревич

# Операционные системы

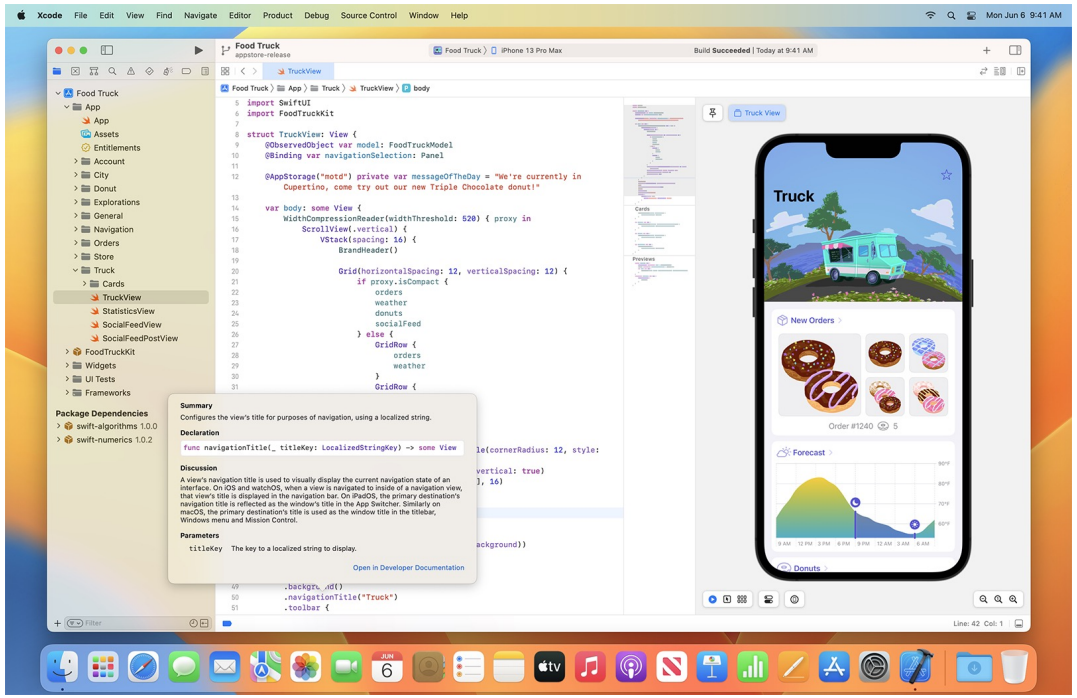
- **Мобильное приложение** ( «Mobile application») — программное изделие, разновидность прикладного программного обеспечения, предназначенная для работы на смартфонах, планшетах и других мобильных (портативных, переносных, карманных) устройствах


# Языки

- iOS: Objective-C, Swift
- Android: Java, Kotlin

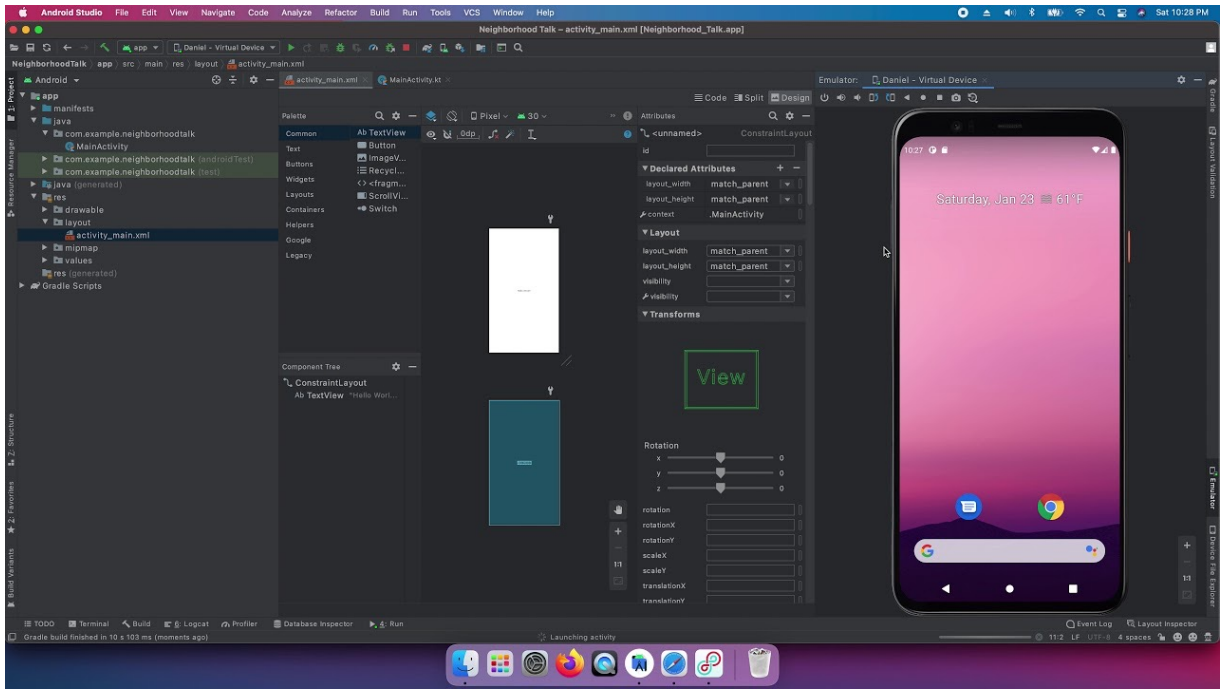


# Среды разработки

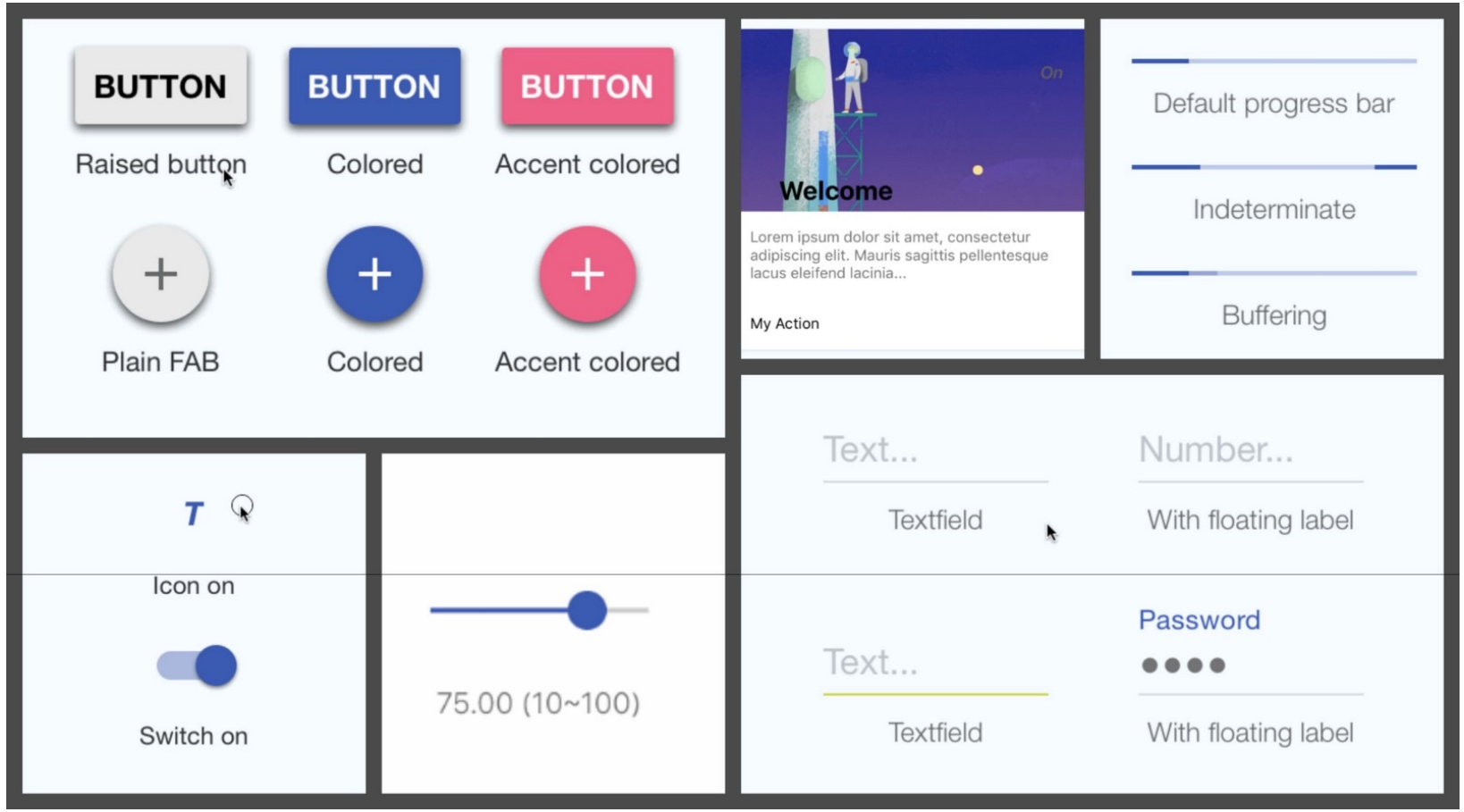



- Xcode

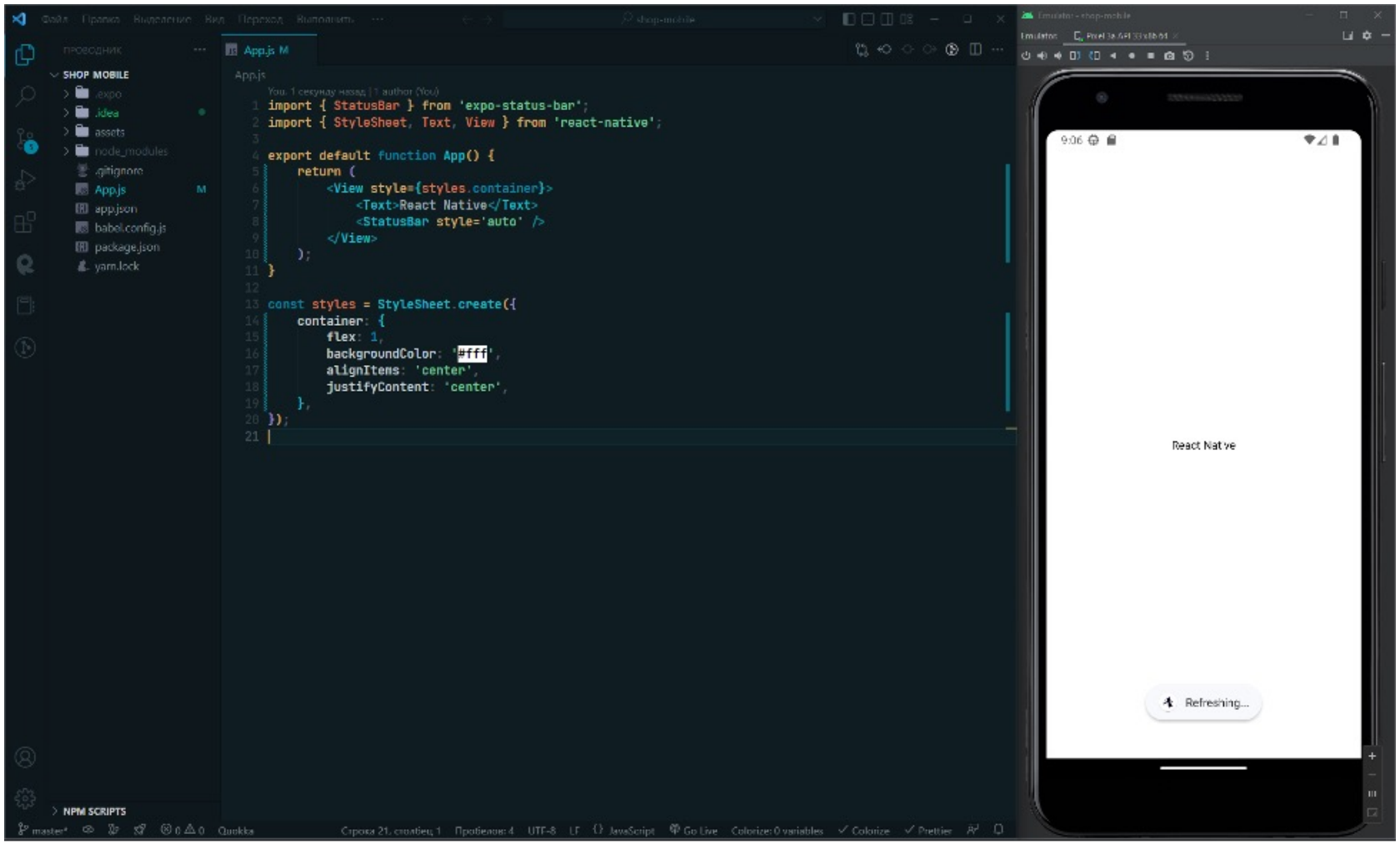
- Android Studio



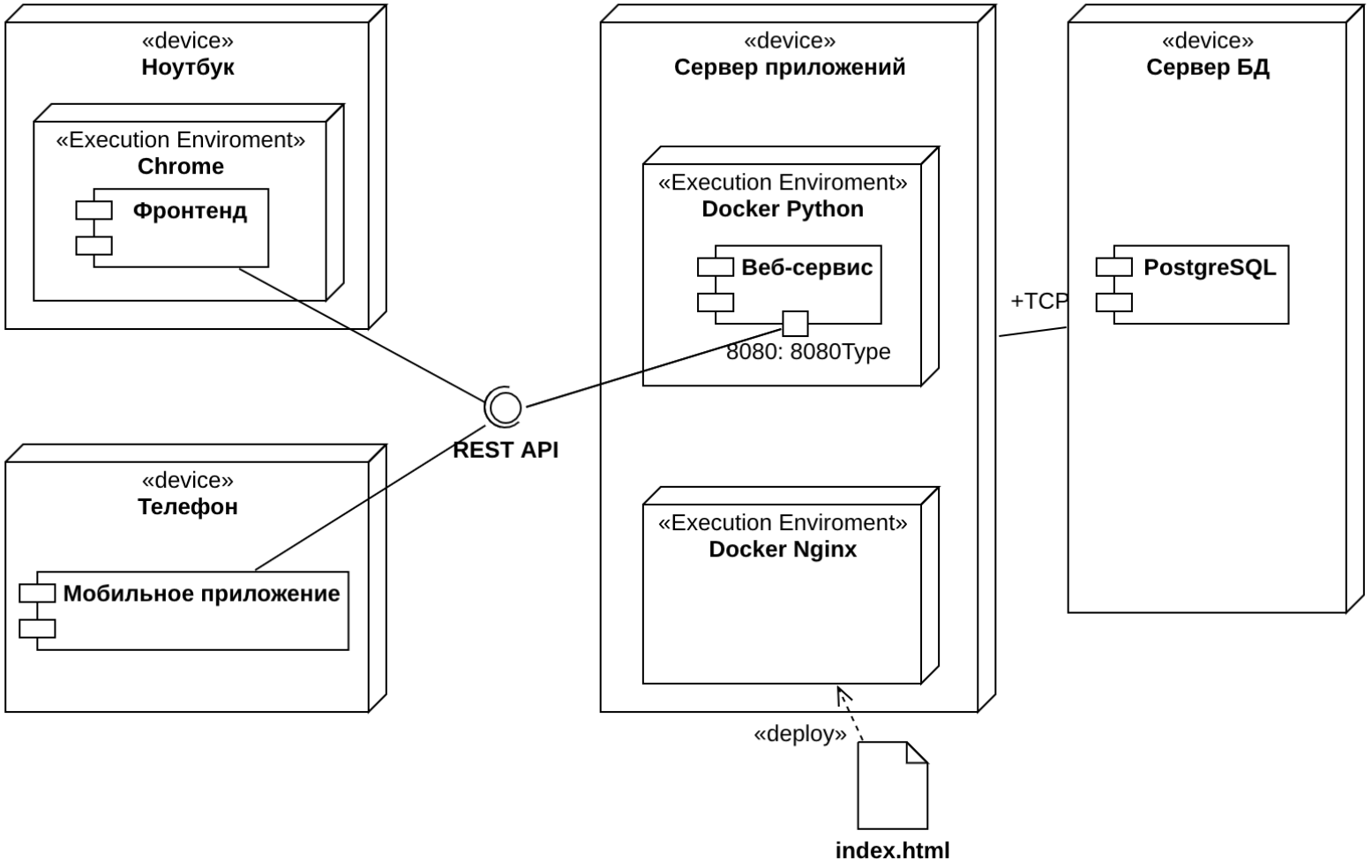
# React Native



# React Native



# Трехзвенная архитектура. API



# Модель данных

- В данном пункте мы создадим модель данных, которая соответствует тому, что вы уже создали на бэкенде.
- В эту модель данных будет парситься json. Также мы создадим запрос к вашему сервису и сам парсинг ответа.
- Прежде чем приступать к созданию подключения сервиса необходимо задать модель с данными, которые придут в ответе от сервиса.

```
import Foundation

struct WeatherData: Codable {
    var location: Location
    var current: Current
}

struct Location: Codable {
    var name: String
    var country: String
    var region: String
}

struct Current: Codable {
    var observation_time: String
    var temperature: Int
    var wind_speed: Int
    var pressure: Int
    var feelslike: Int
}
```

# Генерация запроса

```swift
func configureURLRequest(city: String) -> URLRequest {
    var request: URLRequest
    let acsessToken: String = "b849bbbe085e655065bb8546ec2a8dd5" // нужен для weather-api

    let queryItems = [
        URLQueryItem(name: "access_key", value: acsessToken),
        URLQueryItem(name: "query", value: "'\(city)'")
    ]
    guard var urlComponents = URLComponents(string: "http://api.weatherstack.com/current") else {
        // если не получится создать компоненты из своих query параметров, то переходим на google
        return URLRequest(url: URL(string: "https://google.com")!)
    }

    urlComponents.queryItems = queryItems

    guard let url = urlComponents.url else {
        // если не получится создать url из своего адреса, то переходим на google
        return URLRequest(url: URL(string: "https://google.com")!)
    }

    request = URLRequest(url: url)
    request.httpMethod = ApiMethods.post.rawValue // устанавливаем метод запроса через enum
    return request
}
```

# Запросы к API

- Создание запросов к собственному api сервису в отдельном файле

```swift
import Foundation

final class ApiService {

    func getWeatherData(city: String, completion: @escaping (WeatherData?, Error?)-> ()) {
        let request = configureURLRequest(city: city) // конфигурация кастомного запроса

        URLSession.shared.dataTask(with: request, completionHandler: { data, response, error in  // completionHa

            if let error = error {
                print("error")
                completion(nil, error)
            }
            if let response = response {
                print(response)
            }
            guard let data = data else {
                completion(nil, error)
                return
            }

            do {
                let weatherData = try JSONDecoder().decode(WeatherData.self, from: data) //декодируем json в соз
                completion(weatherData, nil)
            } catch let error {
                completion(nil, error)
            }
        }).resume() // запускаем задачу
    }
}
```

# Заполнение страницы данными

```
private func loadWeatherData(cities: [String]) {
        guard let apiService = apiService else { // раскрытие опциональной переменной apiService
            return
        }

        cities.forEach {
            apiService.getWeatherData(city: $0, completion: { [weak self] (weatherData, error) in // weak self дл
                DispatchQueue.main.async { // запуск асинхронной задачи на main потоке из-за обработки на ui !!!
                    guard let self = self else { return }
                    if let error = error {
     // показ ошибки
                        self.present(UIAlertController(title: "ERROR", message: error.localizedDescription, pref
                        return
                    }
                    if let weatherData = weatherData {
                        self.weatherListData.append(weatherData) // массив с данными о погоде
                    }
                    self.weatherListTableView.reloadData() // перезагрузка таблицы для отображения новых данных
                }
            })
        }
    }
```

# Переходы между страницами

- Далее необходимо добавить переход на данный экран с основного

```
import Foundation
import UIKit

final class WeatherInfoViewController: UIViewController {
    private var weatherData: WeatherData

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    init(weatherData: WeatherData) {
        self.weatherData = weatherData
        super.init(nibName: nil, bundle: nil)
    }
}
```

```
func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        let weatherInfoViewController = WeatherInfoViewController(weatherData: self.weatherListData[indexPath.row])
        navigationController?.pushViewController(weatherInfoViewController, animated: true)
    }
```

# Заполнение детальной информации

- Создадим функцию, которая будет сохранять в текстовые лейблы значения строк с детальной информацией об объекте, которые мы передали с первого экрана.
- которая вызывается из инициализатора контроллера

```
private var weatherData: WeatherData

init(weatherData: WeatherData) {
        self.weatherData = weatherData
        super.init(nibName: nil, bundle: nil)
        fillData(withModel: weatherData)
    }
```

```
func fillData(withModel: WeatherData) {
      degreeLabel.text =  "Temperature: " + String(withModel.current.temperature)
      windLabel.text = "Wind speed: " + String(withModel.current.wind_speed)
      pressureLabel.text = "Pressure: " + String(withModel.current.pressure)
      feelslikeLabel.text = "Feels like: " + String(withModel.current.feelslike)
  }
```

# Верстка страницы с деталями

```swift
final class WeatherInfoViewController: UIViewController {
//добавим на экран элементы, которые хотим отобразить на экране
    private let imageView = UIImageView()
    private let degreeLabel = UILabel()
    private let windLabel = UILabel()
    private let pressureLabel = UILabel()
    private let feelslikeLabel = UILabel()

//создадим переменную для хранения детальной информации об объекте
    private var weatherData: WeatherData

    override func viewDidLoad() {
        super.viewDidLoad()
        configure()
        configureDataElements()
    }
```

```swift
//зададим базовые настройки для текстовых полей и добавим их на экран
    private func configureDataElements() {
        [degreeLabel, windLabel, pressureLabel, feelslikeLabel].forEach {
            $0.translatesAutoresizingMaskIntoConstraints = false
            $0.font = UIFont.systemFont(ofSize: 20, weight: .bold)
            $0.textColor = .white
            view.addSubview($0)
        }

//зададим констрейнты и базовые настройки для картинки
        imageView.translatesAutoresizingMaskIntoConstraints = false
        view.addSubview(imageView)
        imageView.heightAnchor.constraint(equalToConstant: 250).isActive = true
        imageView.widthAnchor.constraint(equalToConstant: 200).isActive = true
        imageView.leftAnchor.constraint(equalTo: view.leftAnchor, constant: 5).isActive = true
        imageView.topAnchor.constraint(equalTo: view.safeAreaLayoutGuide.topAnchor).isActive = true

        imageView.image = UIImage(named: "sunny")
```