Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Керівник роботи

\_\_\_\_\_\_\_\_ Євгеній ВОВК

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 р.

**Мобільний застосунок «метеорологічна карта»**

**Текст програми**

КПІ.ІТ-0115.045440.03.12

“ПОГОДЖЕНО”

Керівник роботи:

\_\_\_\_\_\_\_\_\_\_\_\_ Євгеній ВОВК

|  |  |
| --- | --- |
| Консультант: | Виконавець: |
| \_\_\_\_\_\_\_\_\_\_\_ Максим ГОЛОВЧЕНКО | \_\_\_\_\_\_\_\_\_\_ Олексій КУРКІН |

Київ – 2022

**Файл AppDelegate.swift**

**import** UIKit

**import** RealmSwift

**import** Firebase

**import** FirebaseCore

**import** FirebaseAnalytics

**import** FirebaseCrashlytics

**import** FirebaseRemoteConfig

**import** FirebaseRemoteConfigSwift

**@main**

**class** AppDelegate: UIResponder, UIApplicationDelegate {

**func** application(\_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: **Any**]?) -> Bool {

#if DEBUG

**if** **let** realmFileUrl = Realm.Configuration.defaultConfiguration.fileURL?.absoluteString {

print("Realm file: \(realmFileUrl)")

}

#endif

FirebaseApp.configure()

WeatherAnalytics().logAppLaunched()

**let** remoteConfig = RemoteConfig.remoteConfig()

**let** settings = RemoteConfigSettings()

settings.minimumFetchInterval = 0

remoteConfig.configSettings = settings

**return** **true**

}

// **MARK: UISceneSession Lifecycle**

**func** application(\_ application: UIApplication, configurationForConnecting connectingSceneSession: UISceneSession, options: UIScene.ConnectionOptions) -> UISceneConfiguration {

**return** UISceneConfiguration(name: "Default Configuration", sessionRole: connectingSceneSession.role)

}

}

**Файл SceneDelegate.swift**

**import** UIKit

**class** SceneDelegate: UIResponder, UIWindowSceneDelegate {

**var** window: UIWindow?

**func** scene(\_ scene: UIScene, willConnectTo session: UISceneSession, options connectionOptions: UIScene.ConnectionOptions) {

**guard** **let** \_ = (scene **as**? UIWindowScene) **else** { **return** }

}

}

**Файл Router.swift**

**import** Foundation

**import** UIKit

**enum** Route {

**case** back

**case** weatherDetails

**case** banner

}

**protocol** Router {

**func** route(to route: Route, context: UIViewController, parameter: **Any**?)

}

**extension** Router {

**func** route(to route: Route, context: UIViewController) {

**self**.route(to: route, context: context, parameter: **nil**)

}

}

**Файл Analytics.swift**

**import** Foundation

**import** Firebase

**import** FirebaseAnalytics

**class** WeatherAnalytics {

**func** logAppLaunched() {

Analytics.logEvent("APP\_LAUNCHED", parameters: **nil**)

}

**func** logMapAppeared() {

Analytics.logEvent("MAP\_APPEARED", parameters: **nil**)

}

**func** logOpenCityDetailsEvent(city: String) {

Analytics.logEvent("OPEN\_CITY\_DETAILS", parameters: ["City": city])

}

**func** logFetchCity(on location: String) {

Analytics.logEvent("FETCH\_CITY", parameters: ["Location": location])

}

**func** logFetchWeather(on location: String) {

Analytics.logEvent("FETCH\_WEATHER", parameters: ["Location": location])

}

}

**Файл MaintenanceModeManager.swift**

**import** Foundation

**import** Firebase

**import** FirebaseRemoteConfig

**import** FirebaseRemoteConfigSwift

**class** MaintenanceModeManager {

**var** maintenanceModeIsEnabled: Bool {

RemoteConfig.remoteConfig().configValue(forKey: "maintenanceMode").boolValue

}

**func** setup() {

**try**? RemoteConfig.remoteConfig().setDefaults(from: ["maintenanceMode": **false**])

RemoteConfig.remoteConfig().fetchAndActivate()

}

}

**Файл City.swift**

**import** Foundation

**import** CoreLocation

**import** SwiftyJSON

**struct** City {

**let** name: String

**let** latitude: Double

**let** longitude: Double

**var** location: CLLocationCoordinate2D {

.init(latitude: latitude, longitude: longitude)

}

**init**?(json: JSON) {

**guard** **let** name = json["name"].string,

**let** latitude = json["latitude"].double,

**let** longitude = json["longitude"].double **else** {

**return** **nil**

}

**self**.init(name: name, latitude: latitude, longitude: longitude)

}

**init**(name: String, latitude: Double, longitude: Double) {

**self**.name = name

**self**.latitude = latitude

**self**.longitude = longitude

}

}

**Файл CityRealmObjects.swift**

**import** Foundation

**import** RealmSwift

**import** CoreLocation

**class** RealmCoordinates: Object {

@Persisted **var** latitude: Double = 0

@Persisted **var** longitude: Double = 0

}

**extension** CLLocationCoordinate2D {

**var** realmObject: RealmCoordinates {

**let** obj = RealmCoordinates()

obj.latitude = latitude

obj.longitude = longitude

**return** obj

}

**init**(realmObject: RealmCoordinates) {

**self**.init(latitude: realmObject.latitude,

longitude: realmObject.longitude)

}

}

**class** RealmCity: Object {

@Persisted **var** name: String = ""

@Persisted **var** latitude: Double = 0

@Persisted **var** longitude: Double = 0

}

**extension** City {

**var** realmObject: RealmCity {

**let** obj = RealmCity()

obj.name = name

obj.latitude = latitude

obj.longitude = longitude

**return** obj

}

**init**(realmObject: RealmCity) {

**self**.init(name: realmObject.name,

latitude: realmObject.latitude,

longitude: realmObject.longitude)

}

}

**Файл RemoteCitiesStorage.swift**

**import** Foundation

**import** Alamofire

**import** RxSwift

**import** SwiftyJSON

**import** CoreLocation

**class** RemoteCitiesStorage {

**private** **let** headers: HTTPHeaders = [

"X-RapidAPI-Key": Keys.cityApiKey,

"X-RapidAPI-Host": "world-geo-data.p.rapidapi.com"

]

**private** **let** baseUrl = URL(string: "https://world-geo-data.p.rapidapi.com/cities/nearby")!

**func** fetch(latitude: Double, longitude: Double, minimumCityPopulation: Int = 50000) -> Observable<[City]> {

**let** parameters: [String: **Any**] = [

"latitude": latitude,

"longitude": longitude,

"radius": 500,

"min\_population": minimumCityPopulation

]

WeatherAnalytics().logFetchCity(on:

CLLocationCoordinate2D(latitude: latitude,

longitude: longitude)

.stringRepresentation)

**return** Observable.create { [**weak** **self**] observer **in**

**guard** **let** self = **self** **else** {

observer.onCompleted()

**return** Disposables.create()

}

AF.request(self.baseUrl,

method: .get,

parameters: parameters,

headers: self.headers)

.validate(statusCode: 200..<300)

.responseData { response **in**

**if** **let** data = response.data, **let** jsonData = **try**? JSON(data: data),

**let** cities = jsonData["cities"].array {

observer.onNext(cities.compactMap({ City(json: $0) }))

}

observer.onCompleted()

}

**return** Disposables.create()

}

}

}

**Файл** LocalCitiesStorage.swift

**import** Foundation

**import** CoreLocation

**import** RealmSwift

**import** RxSwift

**import** RxRelay

**class** LocalCitiesStorage {

**private** **let** realm = **try**! Realm()

**var** fetchedCoordinates = [CLLocationCoordinate2D]()

**var** citiesSteam: Observable<City> { \_citiesStream.asObservable() }

**var** cities = [City]()

**private** **let** \_citiesStream = PublishRelay<City>()

**init**() {

realm.objects(RealmCoordinates.**self**)

.map({ CLLocationCoordinate2D(realmObject: $0) })

.forEach({ fetchedCoordinates.append($0) })

}

**func** fetchCities() {

realm.objects(RealmCity.**self**)

.map({ City(realmObject: $0) })

.forEach({

\_citiesStream.accept($0)

cities.append($0)

})

}

**func** save(coordinate: CLLocationCoordinate2D) {

**let** realm = realm

**let** realmObject = coordinate.realmObject

realm.writeAsync({

realm.add(realmObject)

})

}

**func** save(cities: [City]) {

**let** realm = realm

realm.writeAsync({

cities.forEach({ realm.add($0.realmObject) })

})

}

}

**Файл CitiesService.swift**

**import** Foundation

**import** CoreLocation

**import** RxSwift

**import** RxRelay

**class** CitiesService {

**var** citiesSteam: Observable<City> { \_citiesStream.asObservable() }

**private** **let** \_citiesStream = PublishRelay<City>()

**private** **let** disposeBag = DisposeBag()

**private** **let** remoteStorage = RemoteCitiesStorage()

**private** **let** localStorage = LocalCitiesStorage()

**private** **lazy** **var** fetchedCoordinates = localStorage.fetchedCoordinates

**init**() {

localStorage.citiesSteam.subscribe(onNext: { [**weak** **self**] city **in**

**self**?.\_citiesStream.accept(city)

}).disposed(by: disposeBag)

}

**func** startFetch() {

localStorage.fetchCities()

}

**func** getCities(for location: CLLocationCoordinate2D) {

**guard** isRemoteFetchNeeded(for: location) **else** {

**return**

}

fetchedCoordinates.append(location)

localStorage.save(coordinate: location)

remoteStorage.fetch(latitude: location.latitude, longitude: location.longitude)

.subscribe(onNext: { [**weak** **self**] cities **in**

**let** cities = cities.filter({ [**weak** **self**] city **in**

**guard** **let** self = **self** **else** { **return** **false** }

**return** !self.fetchedCoordinates

.filter({ $0 != location })

.map({ CLLocation(coordinate: $0) })

.map({ $0.distance(from: .init(coordinate: city.location)) })

.contains(where: { $0 < 500000 })

})

**self**?.localStorage.save(cities: cities)

cities.forEach({

**self**?.\_citiesStream.accept($0)

})

})

.disposed(by: disposeBag)

}

**func** localCities(for location: CLLocationCoordinate2D) -> [City] {

localStorage.cities.filter({$0.location.distance(to: location) < 500000 })

}

**private** **func** isRemoteFetchNeeded(for location: CLLocationCoordinate2D) -> Bool {

**guard** !fetchedCoordinates.isEmpty **else** { **return** **true** }

**return** fetchedCoordinates.first(where: {

$0.distance(to: location) < 250000

}) == **nil**

}

}

**Файл WeatherItem.swift**

**import** UIKit

**import** SwiftyJSON

**struct** WeatherItem {

**let** temperature: Int

**let** windSpeed: Double

**let** humidity: Int

**let** iconId: Int

**let** date: Date

**init**(temperature: Int, windSpeed: Double, humidity: Int, iconId: Int, date: Date) {

**self**.temperature = temperature

**self**.windSpeed = windSpeed

**self**.humidity = humidity

**self**.iconId = iconId

**self**.date = date

}

**init**?(json: JSON) {

**guard** **let** temperature = json["main"]["temp"].double,

**let** windSpeed = json["wind"]["speed"].double,

**let** humidity = json["main"]["humidity"].int,

**let** weatherIconId = json["weather"].array?.first?["id"].int?.toWeatherIconId,

**let** date = json["dt"].int **else** {

**return** **nil**

}

**self**.init(temperature: Int(temperature),

windSpeed: windSpeed,

humidity: humidity,

iconId: weatherIconId,

date: .init(timeIntervalSince1970: Double(date)))

}

}

// **MARK: - WeatherItem utils**

**extension** WeatherItem {

**var** icon: UIImage { .weatherIcon(with: iconId) }

}

**fileprivate** **extension** Int {

**var** toWeatherIconId: Int? {

**switch** **self** {

**case** 200..<300:

**return** 23

**case** 501...502:

**return** 3

**case** 300..<600:

**return** 24

**case** 600..<700:

**return** 6

**case** 700..<800:

**return** 29

**case** 800:

**return** 20

**case** 801:

**return** 1

**case** 802...804:

**return** 26

**default**:

**return** **nil**

}

}

}

**Файл CityWeatherModel.swift**

**import** Foundation

**class** CityWeatherModel {

**let** city: String

**var** weatherList: [WeatherItem]

**var** isOutdated: Bool { weatherList.count < 35 }

**init**(city: String, weatherList: [WeatherItem] = [WeatherItem]()) {

**self**.city = city

**self**.weatherList = weatherList

}

}

**Файл WeatherRealmObjects.swift**

**import** Foundation

**import** RealmSwift

**class** RealmWeatherItem: Object {

@Persisted **var** temperature: Int = 0

@Persisted **var** windSpeed: Double = 0

@Persisted **var** humidity: Int = 0

@Persisted **var** iconId: Int = 20

@Persisted **var** date: Double = 0

@Persisted **var** cityName: String = ""

}

**extension** WeatherItem {

**func** realmObject(for city: String) -> RealmWeatherItem {

**let** obj = RealmWeatherItem()

obj.temperature = temperature

obj.windSpeed = windSpeed

obj.humidity = humidity

obj.iconId = iconId

obj.date = date.timeIntervalSince1970

obj.cityName = city

**return** obj

}

**init**(realmObject: RealmWeatherItem) {

**self**.init(temperature: realmObject.temperature,

windSpeed: realmObject.windSpeed,

humidity: realmObject.humidity,

iconId: realmObject.iconId,

date: Date(timeIntervalSince1970: realmObject.date))

}

}

**Файл RemoteWeatherStorage.swift**

**import** Foundation

**import** Alamofire

**import** RxSwift

**import** SwiftyJSON

**import** CoreLocation

**class** RemoteWeatherStorage {

**private** **let** baseUrl = URL(string: "https://api.openweathermap.org/data/2.5/forecast")!

**func** fetch(latitude: Double, longitude: Double) -> Observable<[WeatherItem]> {

**let** parameters: [String: **Any**] = [

"lat": latitude,

"lon": longitude,

"appid": Keys.weatherApiKey,

"units": "metric"

]

WeatherAnalytics().logFetchWeather(on:

CLLocationCoordinate2D(latitude: latitude,

longitude: longitude)

.stringRepresentation)

**return** Observable.create { [**weak** **self**] observer **in**

**guard** **let** self = **self** **else** {

observer.onCompleted()

**return** Disposables.create()

}

AF.request(self.baseUrl,

method: .get,

parameters: parameters)

.validate(statusCode: 200..<300)

.responseData { response **in**

**if** **let** data = response.data, **let** jsonData = **try**? JSON(data: data),

**let** weather = jsonData["list"].array {

observer.onNext(weather.compactMap({ WeatherItem(json: $0) }))

}

observer.onCompleted()

}

**return** Disposables.create()

}

}

}

**Файл LocalWeatherStorage.swift**

**import** Foundation

**import** CoreLocation

**import** RealmSwift

**import** RxSwift

**import** RxRelay

**class** LocalWeatherStorage {

**private** **let** realm = **try**! Realm()

**private** **var** dataToDelete = [RealmWeatherItem]()

**func** weather(for city: String) -> [WeatherItem] {

realm.objects(RealmWeatherItem.**self**)

.where({ $0.cityName == city })

.compactMap({ [**weak** **self**] weatherItem **in**

**if** Date(timeIntervalSince1970: weatherItem.date) < Date() {

**if** **let** self = **self** {

self.dataToDelete.append(weatherItem)

**if** self.dataToDelete.count > 500 {

**let** dataToDelete = dataToDelete

**let** realm = self.realm

self.dataToDelete = []

realm.writeAsync { dataToDelete.forEach({ realm.delete($0) }) }

}

}

**return** **nil**

}

**return** WeatherItem(realmObject: weatherItem)

})

}

**func** save(weather: [WeatherItem], for city: String) {

**let** realm = realm

realm.writeAsync({

weather.forEach({ realm.add($0.realmObject(for: city)) })

})

}

}

**Файл WeatherService.swift**

**import** Foundation

**import** CoreLocation

**import** RxSwift

**import** RxRelay

**class** WeatherService {

**var** weatherPinCreationStream: Observable<CLLocationCoordinate2D> {

\_weatherPinCreationStream.compactMap({ $0 }).asObservable()

}

**private** **let** \_weatherPinCreationStream = BehaviorRelay<CLLocationCoordinate2D?>(value: **nil**)

**private** **var** weather: [String: CityWeatherModel] = [:]

**private** **let** remoteWeatherStorage = RemoteWeatherStorage()

**private** **let** localWeatherStorage = LocalWeatherStorage()

**private** **let** disposeBag = DisposeBag()

**private** **let** isRefreshEnabled: Bool

**init**() {

**let** appWasEverLauched = UserDefaults.standard.bool(forKey: "appWasEverLauched")

UserDefaults.standard.set(**true**, forKey: "appWasEverLauched")

isRefreshEnabled = appWasEverLauched

}

**func** data(for location: CLLocationCoordinate2D) -> WeatherMapPinData? {

**guard** **let** weatherItem = weather[location.stringRepresentation]?.weatherList.first **else** {

**return** **nil**

}

**return** .init(icon: weatherItem.icon, temperature: weatherItem.temperature)

}

**func** cityWeather(for location: CLLocationCoordinate2D) -> CityWeatherModel? {

**return** weather[location.stringRepresentation]

}

**func** fetchWeather(for city: City) {

**let** localWeather = localWeatherStorage.weather(for: city.name)

**if** !localWeather.isEmpty {

**self**.weather[city.location.stringRepresentation] = .init(city: city.name,

weatherList: localWeather)

**self**.\_weatherPinCreationStream.accept(city.location)

**return**

}

remoteWeatherStorage

.fetch(latitude: city.latitude,

longitude: city.longitude)

.subscribe(onNext: { [**weak** **self**] weatherList **in**

**guard** **let** self = **self** **else** { **return** }

self.weather[city.location.stringRepresentation] = .init(city: city.name,

weatherList: weatherList)

self.localWeatherStorage.save(weather: weatherList, for: city.name)

self.\_weatherPinCreationStream.accept(city.location)

}).disposed(by: **self**.disposeBag)

}

**func** refreshData(for cities: [City]) {

**guard** isRefreshEnabled **else** { **return** }

cities.forEach({ city **in**

**if** **let** weather = weather[city.location.stringRepresentation],

weather.isOutdated {

remoteWeatherStorage

.fetch(latitude: city.latitude,

longitude: city.longitude)

.subscribe(onNext: { [**weak** **self**] weatherList **in**

**guard** **let** self = **self**,

**let** cityWeatherList = self.weather[

city.location.stringRepresentation]?.weatherList

.sorted(by: { $0.date < $1.date }) **else** {

**return**

}

**if** **let** lastItem = cityWeatherList.last {

**let** itemsToAdd = weatherList.filter({ $0.date > lastItem.date })

self.weather[city.location.stringRepresentation]?

.weatherList

.append(contentsOf: itemsToAdd)

self.localWeatherStorage.save(weather: itemsToAdd, for: city.name)

} **else** {

self.weather[city.location.stringRepresentation]?.weatherList = weatherList

self.localWeatherStorage.save(weather: weatherList, for: city.name)

}

}).disposed(by: **self**.disposeBag)

}

})

}

}

**Файл MapViewController.swift**

**import** UIKit

**import** MapKit

**import** RxSwift

**import** RxCocoa

**import** RxRelay

**enum** MapConstants {

**static** **let** defaultCameraZoomRange = MKMapView.CameraZoomRange(minCenterCoordinateDistance: 0,

maxCenterCoordinateDistance: 500000)

}

**class** MapViewController: UIViewController {

**@IBOutlet** **weak** **var** mapView: MKMapView!

**private** **let** viewModel: MapViewModelProtocol = MapViewModel()

**private** **let** locationManager = LocationManager()

**private** **let** router: Router = MapRouter()

**private** **let** disposeBag = DisposeBag()

**override** **func** viewDidLoad() {

**super**.viewDidLoad()

setupMapView()

bindToViewModel()

viewModel.viewDidLoad()

viewModel.onMaintenanceModeEnabled = { [**weak** **self**] **in**

**self**?.showMaintenanceAlert()

}

}

**override** **func** viewDidAppear(\_ animated: Bool) {

**super**.viewDidAppear(animated)

WeatherAnalytics().logMapAppeared()

locationManager.requestAuthorization { [**weak** **self**] location **in**

**self**?.mapView.setCenter(location, animated: **true**)

}

}

**private** **func** bindToViewModel() {

viewModel.weatherPinCreationStream

.subscribe(on: MainScheduler.instance)

.observe(on: MainScheduler.instance)

.subscribe(onNext: { [**weak** **self**] location **in**

**let** pin = MKPointAnnotation()

pin.coordinate = location

**self**?.mapView.addAnnotation(pin)

}).disposed(by: disposeBag)

}

**private** **func** setupMapView() {

mapView.delegate = **self**

mapView.showsUserLocation = **true**

mapView.setCameraZoomRange(MapConstants.defaultCameraZoomRange, animated: **false**)

mapView.register(UserLocationView.**self**)

mapView.register(WeatherPinView.**self**)

}

**private** **func** showMaintenanceAlert() {

**let** alert = UIAlertController(title: "Maintenance mode is Enabled",

message: "Sorry, app is not working for now...(",

preferredStyle: .alert)

alert.addAction(.init(title: "Ok", style: .cancel, handler: { [**weak** **self**] \_ **in**

**self**?.showMaintenanceAlert()

}))

present(alert, animated: **true**)

}

}

// **MARK: - MKMapViewDelegate**

**extension** MapViewController: MKMapViewDelegate {

**func** mapView(\_ mapView: MKMapView, viewFor annotation: MKAnnotation) -> MKAnnotationView? {

**if** annotation **is** MKUserLocation {

**return** mapView.dequeueReusableView(type: UserLocationView.**self**)

}

**guard** **let** weatherPinData = viewModel.data(for: annotation.coordinate) **else** {

**return** **nil**

}

**let** weatherView = mapView.dequeueReusableView(type: WeatherPinView.**self**)

weatherView.setup(temperature: weatherPinData.temperature,

weatherIcon: weatherPinData.icon)

weatherView.onTap = { [**weak** **self**] location **in**

**guard** **let** self = **self** **else** { **return** }

self.router.route(to: .weatherDetails,

context: self,

parameter: self.viewModel.cityWeather(for: location))

}

**return** weatherView

}

**func** mapViewDidChangeVisibleRegion(\_ mapView: MKMapView) {

viewModel.mapCenter.accept(mapView.centerCoordinate)

}

}

**Файл UserLocationView.swift**

**import** UIKit

**import** MapKit

**class** UserLocationView: MKAnnotationView {

**private** **let** locationImageView: UIImageView = {

**let** imageView = withAutoloyaut(UIImageView(image: .location))

imageView.contentMode = .scaleAspectFit

**return** imageView

}()

**override** **init**(annotation: MKAnnotation?, reuseIdentifier: String?) {

**super**.init(annotation: annotation, reuseIdentifier: reuseIdentifier)

frame = .init(x: 0, y: 0, width: 30, height: 30)

centerOffset = .init(x: 0, y: -frame.size.height / 2)

canShowCallout = **false**

addSubview(locationImageView)

locationImageView.pinToSuperview()

}

**required** **init**?(coder aDecoder: NSCoder) {

fatalError("init(coder:) has not been implemented")

}

}

**Файл WeatherPinView.swift**

**import** UIKit

**import** MapKit

**class** WeatherPinView: MKAnnotationView {

**var** onTap: ((CLLocationCoordinate2D)->())?

**private** **let** pinImageView: UIImageView = {

**let** imageView = withAutoloyaut(UIImageView(image: .mapPin))

imageView.contentMode = .scaleAspectFit

**return** imageView

}()

**private** **let** temperatureLabel: UILabel = {

**let** label = withAutoloyaut(UILabel())

label.textAlignment = .center

label.font = .interBold(size: 14)

label.adjustsFontSizeToFitWidth = **true**

**return** label

}()

**private** **let** weatherImageView: UIImageView = {

**let** imageView = withAutoloyaut(UIImageView(image: .sunWithCloudsWeatherIcon))

imageView.contentMode = .scaleAspectFit

**return** imageView

}()

**override** **init**(annotation: MKAnnotation?, reuseIdentifier: String?) {

**super**.init(annotation: annotation, reuseIdentifier: reuseIdentifier)

frame = .init(x: 0, y: 0, width: 50, height: 65)

centerOffset = .init(x: 0, y: -frame.size.height / 2)

canShowCallout = **false**

addSubview(pinImageView)

pinImageView.addSubview(temperatureLabel)

pinImageView.addSubview(weatherImageView)

setupConstraints()

**let** gestureRecognizer = UITapGestureRecognizer(target: **self**, action: **#selector**(pinTapped))

addGestureRecognizer(gestureRecognizer)

}

**required** **init**?(coder aDecoder: NSCoder) {

fatalError("init(coder:) has not been implemented")

}

**func** setup(temperature: Int, weatherIcon: UIImage) {

temperatureLabel.text = "\(temperature)°"

weatherImageView.image = weatherIcon

}

**private** **func** setupConstraints() {

pinImageView.pinToSuperview()

temperatureLabel.constraintSizeEqual(to: 24)

NSLayoutConstraint.activate([

temperatureLabel.topAnchor.constraint(equalTo: pinImageView.topAnchor,

constant: 4),

temperatureLabel.centerXAnchor.constraint(equalTo: pinImageView.centerXAnchor)

])

weatherImageView.constraintSizeEqual(to: 18)

NSLayoutConstraint.activate([

weatherImageView.topAnchor.constraint(equalTo: temperatureLabel.bottomAnchor,

constant: -2),

weatherImageView.centerXAnchor.constraint(equalTo: pinImageView.centerXAnchor)

])

}

**@objc** **private** **func** pinTapped() {

**if** **let** onTap = onTap, **let** coordinate = annotation?.coordinate {

onTap(coordinate)

}

}

}

**Файл MapViewModel.swift**

**import** Foundation

**import** RxSwift

**import** RxRelay

**import** CoreLocation

**struct** WeatherMapPinData {

**let** icon: UIImage

**let** temperature: Int

}

**protocol** MapViewModelProtocol: AnyObject {

**var** weatherPinCreationStream: Observable<CLLocationCoordinate2D> { **get** }

**var** mapCenter: BehaviorRelay<CLLocationCoordinate2D> { **get** }

**var** onMaintenanceModeEnabled: (()->())? { **get** **set** }

**func** viewDidLoad()

**func** data(for location: CLLocationCoordinate2D) -> WeatherMapPinData?

**func** cityWeather(for location: CLLocationCoordinate2D) -> CityWeatherModel?

}

**class** MapViewModel: MapViewModelProtocol {

**let** mapCenter = BehaviorRelay<CLLocationCoordinate2D>(value: .zero)

**var** weatherPinCreationStream: Observable<CLLocationCoordinate2D> {

weatherService.weatherPinCreationStream

}

**var** onMaintenanceModeEnabled: (()->())? = **nil**

**private** **let** disposeBag = DisposeBag()

**private** **let** citiesSevice = CitiesService()

**private** **let** weatherService = WeatherService()

**private** **let** maintenanceModeManager = MaintenanceModeManager()

**private** **var** lastRefreshedLocation: CLLocationCoordinate2D?

**init**() {

mapCenter.skip(1)

.subscribe(onNext: { [**weak** **self**] location **in**

**guard** **let** self = **self**,

!self.maintenanceModeManager.maintenanceModeIsEnabled **else** {

self?.onMaintenanceModeEnabled?()

self?.onMaintenanceModeEnabled = **nil**

**return**

}

self.citiesSevice.getCities(for: location)

**if** self.lastRefreshedLocation == **nil** ||

self.lastRefreshedLocation!.distance(to: location) > 100000 {

**let** visibleCities = self.citiesSevice.localCities(for: location)

self.weatherService.refreshData(for: visibleCities)

self.lastRefreshedLocation = location

}

}).disposed(by: disposeBag)

citiesSevice.citiesSteam.subscribe(onNext: { [**weak** **self**] city **in**

**guard** **let** self = **self**,

!self.maintenanceModeManager.maintenanceModeIsEnabled **else** {

self?.onMaintenanceModeEnabled?()

self?.onMaintenanceModeEnabled = **nil**

**return**

}

self.weatherService.fetchWeather(for: city)

}).disposed(by: disposeBag)

maintenanceModeManager.setup()

}

**func** viewDidLoad() {

citiesSevice.startFetch()

}

**func** data(for location: CLLocationCoordinate2D) -> WeatherMapPinData? {

weatherService.data(for: location)

}

**func** cityWeather(for location: CLLocationCoordinate2D) -> CityWeatherModel? {

weatherService.cityWeather(for: location)

}

}

**Файл LocationManager.swift**

**import** Foundation

**import** CoreLocation

**class** LocationManager: NSObject {

**private** **var** onGetLocation: ((CLLocationCoordinate2D)->())?

**private** **var** isAutorized = **false**

**private** **let** locationManager = CLLocationManager()

**override** **init**() {

**super**.init()

locationManager.delegate = **self**

}

**func** requestAuthorization(with closure: **@escaping**((CLLocationCoordinate2D)->())) {

**if** !isAutorized {

onGetLocation = closure

locationManager.requestWhenInUseAuthorization()

**return**

}

**if** **let** location = locationManager.location?.coordinate {

closure(location)

}

}

}

**extension** LocationManager: CLLocationManagerDelegate {

**func** locationManagerDidChangeAuthorization(\_ manager: CLLocationManager) {

isAutorized = **true**

**if** **let** location = manager.location?.coordinate {

onGetLocation?(location)

}

}

}

**Файл MapRouter.swift**

**import** UIKit

**class** MapRouter: Router {

**func** route(to route: Route, context: UIViewController, parameter: **Any**?) {

**guard** route == .weatherDetails **else** {

fatalError("MapRouter error: Unknown route")

}

**guard** **let** weatherData = parameter **as**? CityWeatherModel,

**let** viewModel = WeatherDetailsViewModel(cityWeather: weatherData) **else** {

**return**

}

WeatherAnalytics().logOpenCityDetailsEvent(city: viewModel.city)

context.present(WeatherDetailsViewController.initiate(viewModel: viewModel),

animated: **true**)

}

}

**Файл ShortWeatherData.swift**

**import** Foundation

**import** UIKit

**struct** ShortWeatherData {

**let** temperature: Int

**let** iconId: Int

**let** date: Date

**init**(temperature: Int, iconId: Int, date: Date) {

**self**.temperature = temperature

**self**.iconId = iconId

**self**.date = date

}

}

**extension** ShortWeatherData {

**var** icon: UIImage { .weatherIcon(with: iconId) }

**var** hour: String {

**let** dateFormatter = with(DateFormatter()) {

$0.dateFormat = "h:mm"

$0.locale = .init(identifier: "en\_US")

}

**return** dateFormatter.string(from: date)

}

**var** weekDay: String {

**let** dateFormatter = with(DateFormatter(), completion: {

$0.dateFormat = "EEEE"

$0.locale = .init(identifier: "en\_US")

})

**return** dateFormatter.string(from: date)

}

}

// **MARK: - Mapper**

**extension** ShortWeatherData {

**init**(weatherItem: WeatherItem) {

temperature = weatherItem.temperature

iconId = weatherItem.iconId

date = weatherItem.date

}

}

**Файл WeatherDetailsViewController.swift**

**import** UIKit

**extension** WeatherDetailsViewController {

**class** **func** initiate(viewModel: WeatherDetailsViewModelProtocol) -> WeatherDetailsViewController {

**let** storyboard = UIStoryboard(name: "Main", bundle: **nil**)

**let** viewController = storyboard.instantiateViewController(withIdentifier: "WeatherDetailsViewController") **as**! WeatherDetailsViewController

viewController.viewModel = viewModel

viewController.router = WeatherDetailsRouter()

**return** viewController

}

}

**class** WeatherDetailsViewController: UIViewController {

**@IBOutlet** **weak** **var** closeButton: UIImageView!

**@IBOutlet** **weak** **var** bannerImage: UIImageView!

**@IBOutlet** **weak** **var** temperatureLabel: UILabel!

**@IBOutlet** **weak** **var** cityNameLabel: UILabel!

**@IBOutlet** **weak** **var** todayWeatherDetailsView: UIView!

**@IBOutlet** **weak** **var** humidityLabel: UILabel!

**@IBOutlet** **weak** **var** windSpeedLabel: UILabel!

**@IBOutlet** **weak** **var** todayDateLabel: UILabel!

**@IBOutlet** **weak** **var** todayWeatherView: UIView!

**@IBOutlet** **weak** **var** forecastView: UIView!

**@IBOutlet** **weak** **var** todayWeatherHeightConstraint: NSLayoutConstraint!

**fileprivate** **var** todayWeatherViewHeight: CGFloat {

todayWeatherHeightConstraint.constant

}

**fileprivate** **var** viewModel: WeatherDetailsViewModelProtocol!

**fileprivate** **var** router: Router!

**fileprivate** **var** forecastTableViewManager: WeatherForecastTableManager!

**fileprivate** **var** todayWeatherCollectionViewManager: TodayWeatherCollectionViewManager!

**override** **func** viewDidLoad() {

**super**.viewDidLoad()

setupSubviews()

setData()

}

**override** **func** viewDidAppear(\_ animated: Bool) {

**super**.viewDidAppear(animated)

forecastTableViewManager.setupScroll(for: forecastView.bounds.height)

}

**@objc** **private** **func** onCloseButtonTapped() {

router.route(to: .back, context: **self**)

}

**@objc** **private** **func** onBannerTapped() {

router.route(to: .banner, context: **self**)

}

}

// **MARK: - Subviews**

**private** **extension** WeatherDetailsViewController {

**func** setupSubviews() {

todayWeatherDetailsView.layer.cornerRadius = 20

closeButton.addGestureRecognizer(

UITapGestureRecognizer(target: **self**, action: **#selector**(onCloseButtonTapped)))

bannerImage.addGestureRecognizer(

UITapGestureRecognizer(target: **self**, action: **#selector**(onBannerTapped)))

// Today weather collection view

**let** layout = UICollectionViewFlowLayout()

layout.sectionInset = .zero

layout.scrollDirection = .horizontal

layout.minimumInteritemSpacing = 0

layout.minimumLineSpacing = 0

**let** collectionView = withAutoloyaut(UICollectionView(frame: .zero,

collectionViewLayout: layout))

collectionView.isScrollEnabled = **false**

collectionView.showsVerticalScrollIndicator = **false**

collectionView.showsHorizontalScrollIndicator = **false**

collectionView.backgroundColor = .clear

collectionView.allowsSelection = **false**

todayWeatherView.addSubview(collectionView)

collectionView.pinToSuperview()

todayWeatherCollectionViewManager = .init(items: viewModel.todayWeatherForecast,

rowHeight: todayWeatherViewHeight,

collectionView: collectionView)

// Forecast table

**let** tableView = withAutoloyaut(UITableView(frame: .zero))

tableView.separatorStyle = .none

tableView.backgroundColor = .clear

tableView.tableFooterView = UIView()

tableView.showsVerticalScrollIndicator = **false**

tableView.showsHorizontalScrollIndicator = **false**

tableView.isScrollEnabled = **false**

forecastView.addSubview(tableView)

tableView.pinToSuperview()

forecastTableViewManager = .init(forecastItems: viewModel.nextDaysForecast,

tableView: tableView)

}

**func** setData() {

temperatureLabel.text = viewModel.temperature

cityNameLabel.text = viewModel.city

humidityLabel.text = viewModel.humidity

windSpeedLabel.text = viewModel.windSpeed

todayDateLabel.text = viewModel.todayDate

}

}

**Файл TodayWeatherCollectionViewManager.swift**

**import** Foundation

**import** UIKit

**class** TodayWeatherCollectionViewManager: NSObject, UICollectionViewDelegate, UICollectionViewDataSource, UICollectionViewDelegateFlowLayout {

**private** **let** items: [ShortWeatherData]

**private** **let** rowHeight: CGFloat

**private** **weak** **var** collectionView: UICollectionView?

**init**(items: [ShortWeatherData], rowHeight: CGFloat, collectionView: UICollectionView) {

**self**.items = items

**self**.rowHeight = rowHeight

**self**.collectionView = collectionView

**super**.init()

collectionView.register(TodayWeatherCollectionViewCell.**self**,

forCellWithReuseIdentifier: TodayWeatherCollectionViewCell.identifier)

collectionView.delegate = **self**

collectionView.dataSource = **self**

}

**func** collectionView(\_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {

items.count

}

**func** collectionView(\_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) -> UICollectionViewCell {

**guard** **let** cell = collectionView

.dequeueReusableCell(withReuseIdentifier: TodayWeatherCollectionViewCell .identifier,

for: indexPath) **as**? TodayWeatherCollectionViewCell **else** {

**return** UICollectionViewCell()

}

cell.setup(with: items[indexPath.row])

**return** cell

}

**func** collectionView(\_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout, sizeForItemAt indexPath: IndexPath) -> CGSize {

.init(width: collectionView.bounds.width / items.count.cgFloat, height: rowHeight)

}

}

**Файл TodayWeatherCollectionViewCell.swift**

**import** UIKit

**class** TodayWeatherCollectionViewCell: UICollectionViewCell {

**static** **let** identifier = "TodayWeatherCollectionViewCell"

**private** **let** temperatureLabel: UILabel = {

**let** label = withAutoloyaut(UILabel())

label.textAlignment = .right

label.font = .interRegular(size: 18)

**return** label

}()

**private** **let** weatherIconImageView: UIImageView = {

**let** imageView = withAutoloyaut(UIImageView(frame: .zero))

imageView.contentMode = .scaleAspectFill

**return** imageView

}()

**private** **let** timeLabel: UILabel = {

**let** label = withAutoloyaut(UILabel())

label.textAlignment = .right

label.font = .interRegular(size: 18)

**return** label

}()

**override** **init**(frame: CGRect) {

**super**.init(frame: frame)

backgroundColor = .clear

contentView.backgroundColor = .clear

contentView.addSubview(temperatureLabel)

contentView.addSubview(weatherIconImageView)

contentView.addSubview(timeLabel)

addConstraints()

}

**required** **init**?(coder: NSCoder) {

fatalError("init(coder:) has not been implemented")

}

**func** setup(with data: ShortWeatherData) {

temperatureLabel.text = "\(data.temperature)ºC"

weatherIconImageView.image = data.icon

timeLabel.text = data.hour

}

**private** **func** addConstraints() {

NSLayoutConstraint.activate([

temperatureLabel.topAnchor.constraint(equalTo: contentView.topAnchor,

constant: 14),

temperatureLabel.centerXAnchor.constraint(equalTo: contentView.centerXAnchor)

])

weatherIconImageView.constraintSizeEqual(to: 35)

NSLayoutConstraint.activate([

weatherIconImageView.centerYAnchor.constraint(equalTo: contentView.centerYAnchor),

weatherIconImageView.centerXAnchor.constraint(equalTo: contentView.centerXAnchor)

])

NSLayoutConstraint.activate([

timeLabel.bottomAnchor.constraint(equalTo: contentView.bottomAnchor,

constant: -14),

timeLabel.centerXAnchor.constraint(equalTo: contentView.centerXAnchor)

])

}

}

**Файл WeatherForecastTableManager.swift**

**import** Foundation

**import** UIKit

**class** WeatherForecastTableManager: NSObject, UITableViewDelegate, UITableViewDataSource {

**static** **let** rowHeight: CGFloat = 40

**private** **let** forecastItems: [ShortWeatherData]

**private** **weak** **var** tableView: UITableView?

**init**(forecastItems: [ShortWeatherData], tableView: UITableView) {

**self**.forecastItems = forecastItems

**self**.tableView = tableView

**super**.init()

tableView.register(WeatherForecastTableViewCell.**self**,

forCellReuseIdentifier: WeatherForecastTableViewCell.identifier)

tableView.delegate = **self**

tableView.dataSource = **self**

}

**func** reloadData() {

tableView?.reloadData()

}

**func** setupScroll(for height: CGFloat) {

**let** contentHeight = WeatherForecastTableManager.rowHeight \* forecastItems.count.cgFloat

tableView?.isScrollEnabled = contentHeight > height

}

**func** tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

**return** forecastItems.count

}

**func** tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

**guard** **let** cell = tableView.dequeueReusableCell(withIdentifier:

WeatherForecastTableViewCell.identifier)

**as**? WeatherForecastTableViewCell **else** {

**return** UITableViewCell()

}

cell.setup(with: forecastItems[indexPath.row])

**return** cell

}

**func** tableView(\_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {

WeatherForecastTableManager.rowHeight

}

}

**Файл WeatherForecastTableViewCell.swift**

**import** UIKit

**class** WeatherForecastTableViewCell: UITableViewCell {

**static** **let** identifier = "WeatherForecastTableViewCell"

**private** **let** dayLabel: UILabel = {

**let** label = withAutoloyaut(UILabel())

label.textAlignment = .left

label.font = .interMedium(size: 18)

**return** label

}()

**private** **let** weatherIconImageView: UIImageView = {

**let** imageView = withAutoloyaut(UIImageView(frame: .zero))

imageView.contentMode = .scaleAspectFill

**return** imageView

}()

**private** **let** temperatureLabel: UILabel = {

**let** label = withAutoloyaut(UILabel())

label.textAlignment = .right

label.font = .interRegular(size: 18)

**return** label

}()

**override** **init**(style: UITableViewCell.CellStyle, reuseIdentifier: String?) {

**super**.init(style: style, reuseIdentifier: reuseIdentifier)

selectionStyle = .none

backgroundColor = .clear

contentView.backgroundColor = .clear

contentView.addSubview(dayLabel)

contentView.addSubview(weatherIconImageView)

contentView.addSubview(temperatureLabel)

addConstraints()

}

**required** **init**?(coder: NSCoder) {

fatalError("init(coder:) has not been implemented")

}

**func** setup(with data: ShortWeatherData) {

dayLabel.text = data.weekDay

weatherIconImageView.image = data.icon

temperatureLabel.text = "\(data.temperature)ºC"

}

**private** **func** addConstraints() {

NSLayoutConstraint.activate([

dayLabel.leadingAnchor.constraint(equalTo: contentView.leadingAnchor),

dayLabel.centerYAnchor.constraint(equalTo: contentView.centerYAnchor)

])

weatherIconImageView.constraintSizeEqual(to: 24)

NSLayoutConstraint.activate([

weatherIconImageView.leadingAnchor.constraint(equalTo: contentView.leadingAnchor,

constant: 130),

weatherIconImageView.centerYAnchor.constraint(equalTo: contentView.centerYAnchor)

])

NSLayoutConstraint.activate([

temperatureLabel.trailingAnchor.constraint(equalTo: contentView.trailingAnchor, constant: -10),

temperatureLabel.centerYAnchor.constraint(equalTo: contentView.centerYAnchor)

])

}

}

**Файл WeatherDetailsViewModel.swift**

**import** Foundation

**import** RxSwift

**import** RxRelay

**protocol** WeatherDetailsViewModelProtocol {

**var** isComeBackAliveBannerHidden: BehaviorRelay<Bool> { **get** }

**var** city: String { **get** }

**var** temperature: String { **get** }

**var** humidity: String { **get** }

**var** windSpeed: String { **get** }

**var** todayDate: String { **get** }

**var** todayWeatherForecast: [ShortWeatherData] { **get** }

**var** nextDaysForecast: [ShortWeatherData] { **get** }

}

**class** WeatherDetailsViewModel: WeatherDetailsViewModelProtocol {

**var** isComeBackAliveBannerHidden = BehaviorRelay<Bool>(value: **true**)

**var** city: String { cityWeather.city.uppercased() }

**var** temperature: String { "\(cityWeather.weatherList.first!.temperature)°" }

**var** humidity: String { "\(cityWeather.weatherList.first!.humidity)%" }

**var** windSpeed: String { "\(cityWeather.weatherList.first!.windSpeed) m/s" }

**var** todayDate: String {

todayDateFormatter.string(from: cityWeather.weatherList.first!.date)

}

**lazy** **var** todayWeatherForecast: [ShortWeatherData] = {

**var** itemsCount = cityWeather.weatherList.count > 4 ? 4 : cityWeather.weatherList.count

**return** cityWeather.weatherList[0..<itemsCount].map({ .init(weatherItem: $0) })

}()

**lazy** **var** nextDaysForecast: [ShortWeatherData] = {

forecastFilter(items: cityWeather.weatherList, by: 12)

.map({ .init(weatherItem: $0) })

}()

**private** **let** cityWeather: CityWeatherModel

**private** **let** todayDateFormatter = with(DateFormatter()) {

$0.dateFormat = "MMM, d"

$0.locale = .init(identifier: "en\_US")

}

**private** **let** forecastDateFormatter = with(DateFormatter()) {

$0.dateFormat = "HH"

$0.timeZone = TimeZone(abbreviation: "UTC")

$0.locale = .init(identifier: "en\_US")

}

**init**?(cityWeather: CityWeatherModel) {

**if** cityWeather.weatherList.isEmpty { **return** **nil** }

**self**.cityWeather = cityWeather

}

**private** **func** forecastFilter(items: [WeatherItem], by hour: Int) -> [WeatherItem] {

**return** items.filter({

**if** Calendar.current.isDateInToday($0.date) {

**return** **false**

}

**let** dateHour = forecastDateFormatter.string(from: $0.date)

**return** dateHour == "\(hour)"

})

}

}

**Файл WeatherDetailsRouter.swift**

**import** Foundation

**import** UIKit

**class** WeatherDetailsRouter: Router {

**func** route(to route: Route, context: UIViewController, parameter: **Any**?) {

**if** route == .back {

context.dismiss(animated: **true**)

**return**

}

**if** route == .banner {

**if** **let** url = URL(string: "https://savelife.in.ua/en/") {

UIApplication.shared.open(url)

}

**return**

}

fatalError("MapRouter error: Unknown route")

}

}

**Файл With.swift**

**import** Foundation

**import** UIKit

**func** with<T>(\_ obj: T, completion: **@escaping**(T)->()) -> T {

completion(obj)

**return** obj

}

**func** withAutoloyaut<T: UIView>(\_ view: T) -> T {

view.translatesAutoresizingMaskIntoConstraints = **false**

**return** view

}

**Файл UIView + Constraints.swift**

**import** UIKit

**extension** UIView {

**func** constraintSizeEqual(to constant: CGFloat) {

NSLayoutConstraint.activate([

widthAnchor.constraint(equalToConstant: constant),

heightAnchor.constraint(equalToConstant: constant)

])

}

**func** pinToSuperview() {

**guard** **let** superview = superview **else** { **return** }

NSLayoutConstraint.activate([

topAnchor.constraint(equalTo: superview.topAnchor),

bottomAnchor.constraint(equalTo: superview.bottomAnchor),

leadingAnchor.constraint(equalTo: superview.leadingAnchor),

trailingAnchor.constraint(equalTo: superview.trailingAnchor)

])

}

}

**Файл UIImage + Constants.swift**

**import** UIKit

**extension** UIImage {

**static** **var** mapPin: UIImage { .init(named: "MapPin-Image")! }

**static** **var** location: UIImage { .init(named: "Location-Image")! }

**static** **var** sunWithCloudsWeatherIcon: UIImage { .init(named: "SunWithClouds-Image")! }

**static** **func** weatherIcon(with id: Int) -> UIImage {

**guard** **let** icon = UIImage(named: "Weather-\(id)-icon") **else** {

fatalError("Weather icon with id \(id) not found.")

}

**return** icon

}

}

**Файл MKMapView + Register.swift**

**import** MapKit

**extension** MKMapView {

**func** register<T: MKAnnotationView>(\_: T.**Type**) {

**let** identifier = NSStringFromClass(T.**self**)

register(T.**self**, forAnnotationViewWithReuseIdentifier: identifier)

}

**func** dequeueReusableView<T: MKAnnotationView>(type: T.**Type**) -> T {

**let** identifier = NSStringFromClass(T.**self**)

**guard** **let** view = dequeueReusableAnnotationView(withIdentifier: identifier) **as**? T **else** {

fatalError("Could not dequeue cell with identifier: \(identifier)")

}

**return** view

}

}

**Файл CLLocationCoordinate2D.swift**

**import** MapKit

**extension** CLLocationCoordinate2D {

**static** **var** zero: CLLocationCoordinate2D { .init(latitude: 0, longitude: 0) }

**var** stringRepresentation: String { "\(latitude) \(longitude)" }

**func** distance(to coordinate: CLLocationCoordinate2D) -> Double {

**let** location1 = CLLocation(coordinate: **self**)

**let** location2 = CLLocation(coordinate: coordinate)

**return** location1.distance(from: location2)

}

}

**extension** CLLocation {

**convenience** **init**(coordinate: CLLocationCoordinate2D) {

**self**.init(latitude: coordinate.latitude, longitude: coordinate.longitude)

}

}

**extension** CLLocationCoordinate2D: Equatable {

**public** **static** **func** == (lhs: CLLocationCoordinate2D, rhs: CLLocationCoordinate2D) -> Bool {

lhs.longitude == rhs.longitude && lhs.latitude == rhs.latitude

}

}

**Файл Fonts.swift**

**import** Foundation

**import** UIKit

**extension** UIFont {

**static** **func** interBold(size: CGFloat) -> UIFont {

.init(name: "Inter-Bold", size: size) ?? .systemFont(ofSize: size, weight: .bold)

}

**static** **func** interMedium(size: CGFloat) -> UIFont {

.init(name: "Inter-Medium", size: size) ?? .systemFont(ofSize: size, weight: .medium)

}

**static** **func** interRegular(size: CGFloat) -> UIFont {

.init(name: "Inter-Regular", size: size) ?? .systemFont(ofSize: size, weight: .regular)

}

}

**Файл Int.swift**

**import** Foundation

**extension** Int {

**var** cgFloat: CGFloat { .init(**self**) }

}

**Файл Main.storyboard**

**<?xml version=**"1.0" **encoding=**"UTF-8"**?>**

**<document** type**=**"com.apple.InterfaceBuilder3.CocoaTouch.Storyboard.XIB"version**=**"3.0"toolsVersion**=**"21225"targetRuntime**=**"iOS.CocoaTouch"propertyAccessControl**=**"none"useAutolayout**=**"YES"useTraitCollections**=**"YES"useSafeAreas**=**"YES"colorMatched**=**"YES"initialViewController**=**"BYZ-38-t0r"**>**

**<device** id**=**"retina6\_0"orientation**=**"portrait"appearance**=**"light"**/>**

**<dependencies>**

**<deployment** identifier**=**"iOS"**/>**

**<plugIn** identifier**=**"com.apple.InterfaceBuilder.IBCocoaTouchPlugin"version**=**"21207"**/>**

**<capability** name**=**"Map view configurations"minToolsVersion**=**"14.0"**/>**

**<capability** name**=**"Named colors"minToolsVersion**=**"9.0"**/>**

**<capability** name**=**"Safe area layout guides"minToolsVersion**=**"9.0"**/>**

**<capability** name**=**"System colors in document resources"minToolsVersion**=**"11.0"**/>**

**<capability** name**=**"documents saved in the Xcode 8 format"minToolsVersion**=**"8.0"**/>**

**</dependencies>**

**<customFonts** key**=**"customFonts"**>**

**<array** key**=**"Inter-Bold.ttf"**>**

**<string>**Inter-Bold**</string>**

**</array>**

**<array** key**=**"Inter-Light.ttf"**>**

**<string>**Inter-Light**</string>**

**</array>**

**<array** key**=**"Inter-Regular.ttf"**>**

**<string>**Inter-Regular**</string>**

**</array>**

**</customFonts>**

**<scenes>**

<!--Map View Controller-->

**<scene** sceneID**=**"tne-QT-ifu"**>**

**<objects>**

**<viewController** id**=**"BYZ-38-t0r"customClass**=**"MapViewController"customModule**=**"Weather\_map"customModuleProvider**=**"target"sceneMemberID**=**"viewController"**>**

**<view** key**=**"view"contentMode**=**"scaleToFill"id**=**"8bC-Xf-vdC"**>**

**<rect** key**=**"frame"x**=**"0.0"y**=**"0.0"width**=**"390"height**=**"844"**/>**

**<autoresizingMask** key**=**"autoresizingMask"widthSizable**=**"YES"heightSizable**=**"YES"**/>**

**<subviews>**

**<view** contentMode**=**"scaleToFill"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"BF8-rf-qJH"**>**

**<rect** key**=**"frame"x**=**"0.0"y**=**"0.0"width**=**"390"height**=**"844"**/>**

**<subviews>**

**<mapView** clipsSubviews**=**"YES"multipleTouchEnabled**=**"YES"contentMode**=**"scaleToFill"pitchEnabled**=**"NO"showsPointsOfInterest**=**"NO"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"2Ot-cq-lQm"**>**

**<rect** key**=**"frame"x**=**"0.0"y**=**"0.0"width**=**"390"height**=**"844"**/>**

**<standardMapConfiguration** key**=**"preferredConfiguration"**/>**

**</mapView>**

**</subviews>**

**<color** key**=**"backgroundColor"systemColor**=**"systemBackgroundColor"**/>**

**<constraints>**

**<constraint** firstAttribute**=**"trailing"secondItem**=**"2Ot-cq-lQm"secondAttribute**=**"trailing"id**=**"49B-7f-Lf0"**/>**

**<constraint** firstAttribute**=**"bottom"secondItem**=**"2Ot-cq-lQm"secondAttribute**=**"bottom"id**=**"Un3-dg-VBt"**/>**

**<constraint** firstItem**=**"2Ot-cq-lQm"firstAttribute**=**"top"secondItem**=**"BF8-rf-qJH"secondAttribute**=**"top"id**=**"bvF-ei-wNM"**/>**

**<constraint** firstItem**=**"2Ot-cq-lQm"firstAttribute**=**"leading"secondItem**=**"BF8-rf-qJH"secondAttribute**=**"leading"id**=**"y5U-CG-9x8"**/>**

**</constraints>**

**</view>**

**</subviews>**

**<viewLayoutGuide** key**=**"safeArea"id**=**"6Tk-OE-BBY"**/>**

**<color** key**=**"backgroundColor"systemColor**=**"systemBackgroundColor"**/>**

**<constraints>**

**<constraint** firstItem**=**"BF8-rf-qJH"firstAttribute**=**"trailing"secondItem**=**"6Tk-OE-BBY"secondAttribute**=**"trailing"id**=**"Vje-sQ-5LG"**/>**

**<constraint** firstItem**=**"BF8-rf-qJH"firstAttribute**=**"leading"secondItem**=**"6Tk-OE-BBY"secondAttribute**=**"leading"id**=**"ZtH-wg-b03"**/>**

**<constraint** firstItem**=**"BF8-rf-qJH"firstAttribute**=**"top"secondItem**=**"8bC-Xf-vdC"secondAttribute**=**"top"id**=**"ffS-GT-ntm"**/>**

**<constraint** firstAttribute**=**"bottom"secondItem**=**"BF8-rf-qJH"secondAttribute**=**"bottom"id**=**"riw-jK-DLA"**/>**

**</constraints>**

**</view>**

**<connections>**

**<outlet** property**=**"mapView"destination**=**"2Ot-cq-lQm"id**=**"daI-hw-S1P"**/>**

**</connections>**

**</viewController>**

**<placeholder** placeholderIdentifier**=**"IBFirstResponder"id**=**"dkx-z0-nzr"sceneMemberID**=**"firstResponder"**/>**

**</objects>**

**<point** key**=**"canvasLocation"x**=**"138.46153846153845"y**=**"-2.1327014218009479"**/>**

**</scene>**

<!--Weather Details View Controller-->

**<scene** sceneID**=**"0JF-sO-aZs"**>**

**<objects>**

**<viewController** storyboardIdentifier**=**"WeatherDetailsViewController"id**=**"cTR-qg-7jV"customClass**=**"WeatherDetailsViewController"customModule**=**"Weather\_map"customModuleProvider**=**"target"sceneMemberID**=**"viewController"**>**

**<view** key**=**"view"contentMode**=**"scaleToFill"id**=**"vip-wg-MJg"**>**

**<rect** key**=**"frame"x**=**"0.0"y**=**"0.0"width**=**"390"height**=**"844"**/>**

**<autoresizingMask** key**=**"autoresizingMask"widthSizable**=**"YES"heightSizable**=**"YES"**/>**

**<subviews>**

**<view** contentMode**=**"scaleToFill"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"dly-7P-IIE"**>**

**<rect** key**=**"frame"x**=**"16"y**=**"47"width**=**"358"height**=**"763"**/>**

**<subviews>**

**<imageView** clipsSubviews**=**"YES"contentMode**=**"scaleAspectFit"horizontalHuggingPriority**=**"251"verticalHuggingPriority**=**"251"image**=**"xmark-Image"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"DG4-nx-OKM"**>**

**<rect** key**=**"frame"x**=**"333"y**=**"20"width**=**"25"height**=**"25"**/>**

**<constraints>**

**<constraint** firstAttribute**=**"width"constant**=**"25"id**=**"DeX-31-DTZ"**/>**

**<constraint** firstAttribute**=**"height"constant**=**"25"id**=**"Xub-7R-Qyj"**/>**

**</constraints>**

**</imageView>**

**<imageView** clipsSubviews**=**"YES"contentMode**=**"scaleAspectFit"horizontalHuggingPriority**=**"251"verticalHuggingPriority**=**"251"image**=**"Banner-Image"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"tNd-jS-kdc"**>**

**<rect** key**=**"frame"x**=**"0.0"y**=**"20"width**=**"78"height**=**"40"**/>**

**<constraints>**

**<constraint** firstAttribute**=**"height"constant**=**"40"id**=**"Mix-Ki-wZ1"**/>**

**<constraint** firstAttribute**=**"width"constant**=**"78"id**=**"OYB-nn-Egf"**/>**

**</constraints>**

**</imageView>**

**<label** opaque**=**"NO"userInteractionEnabled**=**"NO"contentMode**=**"left"horizontalHuggingPriority**=**"251"verticalHuggingPriority**=**"251"text**=**"40º"textAlignment**=**"center"lineBreakMode**=**"tailTruncation"baselineAdjustment**=**"alignBaselines"adjustsFontSizeToFit**=**"NO"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"UaS-9i-Yyn"**>**

**<rect** key**=**"frame"x**=**"0.0"y**=**"70"width**=**"358"height**=**"109"**/>**

**<fontDescription** key**=**"fontDescription"name**=**"Inter-Regular"family**=**"Inter"pointSize**=**"90"**/>**

**<nil** key**=**"textColor"**/>**

**<nil** key**=**"highlightedColor"**/>**

**</label>**

**<label** opaque**=**"NO"userInteractionEnabled**=**"NO"contentMode**=**"left"horizontalHuggingPriority**=**"251"verticalHuggingPriority**=**"251"text**=**"MARIUPOL"textAlignment**=**"center"lineBreakMode**=**"tailTruncation"baselineAdjustment**=**"alignBaselines"adjustsFontForContentSizeCategory**=**"YES"adjustsFontSizeToFit**=**"NO"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"zMn-Ha-b91"**>**

**<rect** key**=**"frame"x**=**"0.0"y**=**"189"width**=**"358"height**=**"41.333333333333343"**/>**

**<fontDescription** key**=**"fontDescription"name**=**"Inter-Light"family**=**"Inter"pointSize**=**"34"**/>**

**<nil** key**=**"textColor"**/>**

**<nil** key**=**"highlightedColor"**/>**

**</label>**

**<view** clipsSubviews**=**"YES"contentMode**=**"scaleToFill"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"x1Q-EE-PMJ"**>**

**<rect** key**=**"frame"x**=**"0.0"y**=**"250.33333333333331"width**=**"358"height**=**"49"**/>**

**<subviews>**

**<view** contentMode**=**"scaleToFill"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"QFx-HF-VtE"**>**

**<rect** key**=**"frame"x**=**"20"y**=**"0.0"width**=**"159"height**=**"49"**/>**

**<subviews>**

**<stackView** opaque**=**"NO"contentMode**=**"scaleToFill"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"9zZ-hl-Fyp"**>**

**<rect** key**=**"frame"x**=**"50"y**=**"12.666666666666686"width**=**"59"height**=**"24"**/>**

**<subviews>**

**<imageView** clipsSubviews**=**"YES"userInteractionEnabled**=**"NO"contentMode**=**"scaleAspectFit"horizontalHuggingPriority**=**"251"verticalHuggingPriority**=**"251"image**=**"Humidity-Image"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"fc1-S6-g9A"**>**

**<rect** key**=**"frame"x**=**"0.0"y**=**"0.0"width**=**"24"height**=**"24"**/>**

**<color** key**=**"backgroundColor"white**=**"0.0"alpha**=**"0.0"colorSpace**=**"custom"customColorSpace**=**"genericGamma22GrayColorSpace"**/>**

**<constraints>**

**<constraint** firstAttribute**=**"width"constant**=**"24"id**=**"QhJ-M1-uH9"**/>**

**<constraint** firstAttribute**=**"height"constant**=**"24"id**=**"ewb-5a-RAD"**/>**

**</constraints>**

**</imageView>**

**<label** opaque**=**"NO"userInteractionEnabled**=**"NO"contentMode**=**"left"horizontalHuggingPriority**=**"251"verticalHuggingPriority**=**"251"text**=**"65%"lineBreakMode**=**"tailTruncation"baselineAdjustment**=**"alignBaselines"adjustsFontSizeToFit**=**"NO"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"0y0-Hd-S8d"**>**

**<rect** key**=**"frame"x**=**"24"y**=**"0.0"width**=**"35"height**=**"24"**/>**

**<fontDescription** key**=**"fontDescription"name**=**"Inter-Regular"family**=**"Inter"pointSize**=**"17"**/>**

**<color** key**=**"textColor"name**=**"Background-Color"**/>**

**<nil** key**=**"highlightedColor"**/>**

**</label>**

**</subviews>**

**</stackView>**

**</subviews>**

**<color** key**=**"backgroundColor"white**=**"0.0"alpha**=**"0.0"colorSpace**=**"custom"customColorSpace**=**"genericGamma22GrayColorSpace"**/>**

**<constraints>**

**<constraint** firstItem**=**"9zZ-hl-Fyp"firstAttribute**=**"centerX"secondItem**=**"QFx-HF-VtE"secondAttribute**=**"centerX"id**=**"QHq-QX-Og0"**/>**

**<constraint** firstItem**=**"9zZ-hl-Fyp"firstAttribute**=**"centerY"secondItem**=**"QFx-HF-VtE"secondAttribute**=**"centerY"id**=**"sBV-xV-C3W"**/>**

**</constraints>**

**</view>**

**<view** contentMode**=**"scaleToFill"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"Jyu-IW-fIj"**>**

**<rect** key**=**"frame"x**=**"179"y**=**"0.0"width**=**"179"height**=**"49"**/>**

**<subviews>**

**<stackView** opaque**=**"NO"contentMode**=**"scaleToFill"spacing**=**"2"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"Acn-9a-rgp"**>**

**<rect** key**=**"frame"x**=**"44.666666666666664"y**=**"12.666666666666686"width**=**"89.666666666666686"height**=**"24"**/>**

**<subviews>**

**<imageView** clipsSubviews**=**"YES"userInteractionEnabled**=**"NO"contentMode**=**"scaleAspectFit"horizontalHuggingPriority**=**"251"verticalHuggingPriority**=**"251"image**=**"Wind-Image"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"pC1-zV-fWm"**>**

**<rect** key**=**"frame"x**=**"0.0"y**=**"0.0"width**=**"24"height**=**"24"**/>**

**<color** key**=**"backgroundColor"white**=**"0.0"alpha**=**"0.0"colorSpace**=**"custom"customColorSpace**=**"genericGamma22GrayColorSpace"**/>**

**<constraints>**

**<constraint** firstAttribute**=**"height"constant**=**"24"id**=**"H7U-Lx-L4v"**/>**

**<constraint** firstAttribute**=**"width"constant**=**"24"id**=**"TME-Od-bya"**/>**

**</constraints>**

**</imageView>**

**<label** opaque**=**"NO"userInteractionEnabled**=**"NO"contentMode**=**"left"horizontalHuggingPriority**=**"251"verticalHuggingPriority**=**"251"text**=**"19 km/h"lineBreakMode**=**"tailTruncation"baselineAdjustment**=**"alignBaselines"adjustsFontSizeToFit**=**"NO"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"eAg-Yk-ozH"**>**

**<rect** key**=**"frame"x**=**"26.000000000000025"y**=**"0.0"width**=**"63.666666666666657"height**=**"24"**/>**

**<fontDescription** key**=**"fontDescription"name**=**"Inter-Regular"family**=**"Inter"pointSize**=**"17"**/>**

**<color** key**=**"textColor"name**=**"Background-Color"**/>**

**<nil** key**=**"highlightedColor"**/>**

**</label>**

**</subviews>**

**</stackView>**

**</subviews>**

**<color** key**=**"backgroundColor"white**=**"0.0"alpha**=**"0.0"colorSpace**=**"custom"customColorSpace**=**"genericGamma22GrayColorSpace"**/>**

**<constraints>**

**<constraint** firstItem**=**"Acn-9a-rgp"firstAttribute**=**"centerX"secondItem**=**"Jyu-IW-fIj"secondAttribute**=**"centerX"id**=**"olz-KR-NXf"**/>**

**<constraint** firstItem**=**"Acn-9a-rgp"firstAttribute**=**"centerY"secondItem**=**"Jyu-IW-fIj"secondAttribute**=**"centerY"id**=**"pOE-ts-iU4"**/>**

**</constraints>**

**</view>**

**</subviews>**

**<color** key**=**"backgroundColor"name**=**"SecondaryBackground-Color"**/>**

**<constraints>**

**<constraint** firstItem**=**"QFx-HF-VtE"firstAttribute**=**"leading"secondItem**=**"x1Q-EE-PMJ"secondAttribute**=**"leading"constant**=**"20"id**=**"EzU-yc-obk"**/>**

**<constraint** firstAttribute**=**"height"constant**=**"49"id**=**"LBG-4x-w3x"**/>**

**<constraint** firstItem**=**"QFx-HF-VtE"firstAttribute**=**"top"secondItem**=**"x1Q-EE-PMJ"secondAttribute**=**"top"id**=**"LS6-2a-R5o"**/>**

**<constraint** firstItem**=**"Jyu-IW-fIj"firstAttribute**=**"top"secondItem**=**"x1Q-EE-PMJ"secondAttribute**=**"top"id**=**"SqV-oA-qoR"**/>**

**<constraint** firstAttribute**=**"trailing"secondItem**=**"Jyu-IW-fIj"secondAttribute**=**"trailing"id**=**"VhK-xN-Z0V"**/>**

**<constraint** firstAttribute**=**"bottom"secondItem**=**"Jyu-IW-fIj"secondAttribute**=**"bottom"id**=**"Xkl-4U-9hX"**/>**

**<constraint** firstItem**=**"Jyu-IW-fIj"firstAttribute**=**"leading"secondItem**=**"QFx-HF-VtE"secondAttribute**=**"trailing"id**=**"au9-6e-bHG"**/>**

**<constraint** firstAttribute**=**"bottom"secondItem**=**"QFx-HF-VtE"secondAttribute**=**"bottom"id**=**"lB5-T3-lv8"**/>**

**<constraint** firstAttribute**=**"centerX"secondItem**=**"QFx-HF-VtE"secondAttribute**=**"trailing"id**=**"xZQ-4I-whj"**/>**

**</constraints>**

**</view>**

**<label** opaque**=**"NO"userInteractionEnabled**=**"NO"contentMode**=**"left"horizontalHuggingPriority**=**"251"verticalHuggingPriority**=**"251"text**=**"Today"lineBreakMode**=**"tailTruncation"baselineAdjustment**=**"alignBaselines"adjustsFontSizeToFit**=**"NO"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"szA-Q8-1C6"**>**

**<rect** key**=**"frame"x**=**"0.0"y**=**"319.33333333333331"width**=**"60"height**=**"24.333333333333314"**/>**

**<fontDescription** key**=**"fontDescription"name**=**"Inter-Bold"family**=**"Inter"pointSize**=**"20"**/>**

**<nil** key**=**"textColor"**/>**

**<nil** key**=**"highlightedColor"**/>**

**</label>**

**<label** opaque**=**"NO"userInteractionEnabled**=**"NO"contentMode**=**"left"horizontalHuggingPriority**=**"251"verticalHuggingPriority**=**"251"text**=**"Feb, 24"textAlignment**=**"natural"lineBreakMode**=**"tailTruncation"baselineAdjustment**=**"alignBaselines"adjustsFontSizeToFit**=**"NO"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"hkH-F7-bAm"**>**

**<rect** key**=**"frame"x**=**"294"y**=**"320.66666666666669"width**=**"64"height**=**"22"**/>**

**<fontDescription** key**=**"fontDescription"name**=**"Inter-Regular"family**=**"Inter"pointSize**=**"18"**/>**

**<nil** key**=**"textColor"**/>**

**<nil** key**=**"highlightedColor"**/>**

**</label>**

**<view** contentMode**=**"scaleToFill"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"0E0-LJ-twe"**>**

**<rect** key**=**"frame"x**=**"0.0"y**=**"343.66666666666669"width**=**"358"height**=**"155"**/>**

**<color** key**=**"backgroundColor"white**=**"0.0"alpha**=**"0.0"colorSpace**=**"custom"customColorSpace**=**"genericGamma22GrayColorSpace"**/>**

**<constraints>**

**<constraint** firstAttribute**=**"height"constant**=**"155"id**=**"5np-3E-4Du"**/>**

**</constraints>**

**</view>**

**<label** opaque**=**"NO"userInteractionEnabled**=**"NO"contentMode**=**"left"horizontalHuggingPriority**=**"251"verticalHuggingPriority**=**"251"text**=**"Next Forecast"lineBreakMode**=**"tailTruncation"baselineAdjustment**=**"alignBaselines"adjustsFontSizeToFit**=**"NO"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"Yrr-41-zoR"**>**

**<rect** key**=**"frame"x**=**"0.0"y**=**"508.66666666666657"width**=**"135.66666666666666"height**=**"24.333333333333314"**/>**

**<fontDescription** key**=**"fontDescription"name**=**"Inter-Bold"family**=**"Inter"pointSize**=**"20"**/>**

**<nil** key**=**"textColor"**/>**

**<nil** key**=**"highlightedColor"**/>**

**</label>**

**<imageView** clipsSubviews**=**"YES"userInteractionEnabled**=**"NO"contentMode**=**"scaleAspectFit"horizontalHuggingPriority**=**"251"verticalHuggingPriority**=**"251"image**=**"SmallCalendar-Image"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"4sl-7N-Avw"**>**

**<rect** key**=**"frame"x**=**"338"y**=**"511"width**=**"20"height**=**"20"**/>**

**<color** key**=**"backgroundColor"white**=**"0.0"alpha**=**"0.0"colorSpace**=**"custom"customColorSpace**=**"genericGamma22GrayColorSpace"**/>**

**<constraints>**

**<constraint** firstAttribute**=**"height"constant**=**"20"id**=**"DkU-la-iUx"**/>**

**<constraint** firstAttribute**=**"width"constant**=**"20"id**=**"xbl-1a-SQU"**/>**

**</constraints>**

**</imageView>**

**<view** contentMode**=**"scaleToFill"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"IL3-RD-lUu"**>**

**<rect** key**=**"frame"x**=**"0.0"y**=**"543"width**=**"358"height**=**"220"**/>**

**<color** key**=**"backgroundColor"white**=**"0.0"alpha**=**"0.0"colorSpace**=**"custom"customColorSpace**=**"genericGamma22GrayColorSpace"**/>**

**</view>**

**</subviews>**

**<color** key**=**"backgroundColor"white**=**"0.0"alpha**=**"0.0"colorSpace**=**"custom"customColorSpace**=**"genericGamma22GrayColorSpace"**/>**

**<constraints>**

**<constraint** firstItem**=**"Yrr-41-zoR"firstAttribute**=**"leading"secondItem**=**"dly-7P-IIE"secondAttribute**=**"leading"id**=**"1Oa-Ba-PTA"**/>**

**<constraint** firstItem**=**"hkH-F7-bAm"firstAttribute**=**"centerY"secondItem**=**"szA-Q8-1C6"secondAttribute**=**"centerY"id**=**"1hT-al-Jph"**/>**

**<constraint** firstAttribute**=**"trailing"secondItem**=**"DG4-nx-OKM"secondAttribute**=**"trailing"id**=**"33G-CC-rG7"**/>**

**<constraint** firstItem**=**"0E0-LJ-twe"firstAttribute**=**"top"secondItem**=**"szA-Q8-1C6"secondAttribute**=**"bottom"id**=**"39m-XN-5eC"**/>**

**<constraint** firstItem**=**"UaS-9i-Yyn"firstAttribute**=**"leading"secondItem**=**"dly-7P-IIE"secondAttribute**=**"leading"id**=**"9bb-Fu-Vso"**/>**

**<constraint** firstItem**=**"zMn-Ha-b91"firstAttribute**=**"leading"secondItem**=**"dly-7P-IIE"secondAttribute**=**"leading"id**=**"BoP-n6-OYb"**/>**

**<constraint** firstItem**=**"0E0-LJ-twe"firstAttribute**=**"leading"secondItem**=**"dly-7P-IIE"secondAttribute**=**"leading"id**=**"Crh-qI-fzZ"**/>**

**<constraint** firstAttribute**=**"trailing"secondItem**=**"hkH-F7-bAm"secondAttribute**=**"trailing"id**=**"Cx4-z6-wUh"**/>**

**<constraint** firstItem**=**"UaS-9i-Yyn"firstAttribute**=**"top"secondItem**=**"tNd-jS-kdc"secondAttribute**=**"bottom"constant**=**"10"id**=**"DUS-Mm-MaU"**/>**

**<constraint** firstItem**=**"IL3-RD-lUu"firstAttribute**=**"top"secondItem**=**"Yrr-41-zoR"secondAttribute**=**"bottom"constant**=**"10"id**=**"MIX-gW-3Mq"**/>**

**<constraint** firstItem**=**"zMn-Ha-b91"firstAttribute**=**"top"secondItem**=**"UaS-9i-Yyn"secondAttribute**=**"bottom"constant**=**"10"id**=**"SS9-nu-sOI"**/>**

**<constraint** firstItem**=**"tNd-jS-kdc"firstAttribute**=**"top"secondItem**=**"dly-7P-IIE"secondAttribute**=**"top"constant**=**"20"id**=**"V2X-qi-6Cs"**/>**

**<constraint** firstItem**=**"tNd-jS-kdc"firstAttribute**=**"leading"secondItem**=**"dly-7P-IIE"secondAttribute**=**"leading"id**=**"W6F-5C-pdZ"**/>**

**<constraint** firstAttribute**=**"trailing"secondItem**=**"zMn-Ha-b91"secondAttribute**=**"trailing"id**=**"b3E-Jk-hEZ"**/>**

**<constraint** firstItem**=**"szA-Q8-1C6"firstAttribute**=**"top"secondItem**=**"x1Q-EE-PMJ"secondAttribute**=**"bottom"constant**=**"20"id**=**"bAb-cQ-ODR"**/>**

**<constraint** firstItem**=**"x1Q-EE-PMJ"firstAttribute**=**"top"secondItem**=**"zMn-Ha-b91"secondAttribute**=**"bottom"constant**=**"20"id**=**"jO7-oZ-Pfb"**/>**

**<constraint** firstItem**=**"szA-Q8-1C6"firstAttribute**=**"leading"secondItem**=**"dly-7P-IIE"secondAttribute**=**"leading"id**=**"jTZ-2B-sVd"**/>**

**<constraint** firstAttribute**=**"trailing"secondItem**=**"IL3-RD-lUu"secondAttribute**=**"trailing"id**=**"mMr-mW-m1Q"**/>**

**<constraint** firstAttribute**=**"bottom"secondItem**=**"IL3-RD-lUu"secondAttribute**=**"bottom"id**=**"mn1-NM-hcn"**/>**

**<constraint** firstAttribute**=**"trailing"secondItem**=**"4sl-7N-Avw"secondAttribute**=**"trailing"id**=**"p6b-Qe-Y2R"**/>**

**<constraint** firstItem**=**"Yrr-41-zoR"firstAttribute**=**"top"secondItem**=**"0E0-LJ-twe"secondAttribute**=**"bottom"constant**=**"10"id**=**"pa9-uz-whq"**/>**

**<constraint** firstAttribute**=**"trailing"secondItem**=**"x1Q-EE-PMJ"secondAttribute**=**"trailing"id**=**"qB3-fy-61x"**/>**

**<constraint** firstAttribute**=**"trailing"secondItem**=**"UaS-9i-Yyn"secondAttribute**=**"trailing"id**=**"tIC-d1-Tjm"**/>**

**<constraint** firstItem**=**"4sl-7N-Avw"firstAttribute**=**"centerY"secondItem**=**"Yrr-41-zoR"secondAttribute**=**"centerY"id**=**"uy6-f8-4Qp"**/>**

**<constraint** firstItem**=**"x1Q-EE-PMJ"firstAttribute**=**"leading"secondItem**=**"dly-7P-IIE"secondAttribute**=**"leading"id**=**"ymK-Gx-Xtn"**/>**

**<constraint** firstItem**=**"DG4-nx-OKM"firstAttribute**=**"top"secondItem**=**"dly-7P-IIE"secondAttribute**=**"top"constant**=**"20"id**=**"yuk-ds-yeE"**/>**

**<constraint** firstAttribute**=**"trailing"secondItem**=**"0E0-LJ-twe"secondAttribute**=**"trailing"id**=**"zJb-mA-Dhg"**/>**

**<constraint** firstItem**=**"IL3-RD-lUu"firstAttribute**=**"leading"secondItem**=**"dly-7P-IIE"secondAttribute**=**"leading"id**=**"zb2-Se-wtL"**/>**

**</constraints>**

**</view>**

**</subviews>**

**<viewLayoutGuide** key**=**"safeArea"id**=**"p6i-zt-pjw"**/>**

**<color** key**=**"backgroundColor"name**=**"Background-Color"**/>**

**<constraints>**

**<constraint** firstItem**=**"dly-7P-IIE"firstAttribute**=**"top"secondItem**=**"p6i-zt-pjw"secondAttribute**=**"top"id**=**"KMw-8a-5Xu"**/>**

**<constraint** firstItem**=**"p6i-zt-pjw"firstAttribute**=**"bottom"secondItem**=**"dly-7P-IIE"secondAttribute**=**"bottom"id**=**"bxw-sR-PPs"**/>**

**<constraint** firstItem**=**"dly-7P-IIE"firstAttribute**=**"leading"secondItem**=**"vip-wg-MJg"secondAttribute**=**"leadingMargin"id**=**"eXL-ts-bSS"**/>**

**<constraint** firstAttribute**=**"trailingMargin"secondItem**=**"dly-7P-IIE"secondAttribute**=**"trailing"id**=**"oTs-9N-vEY"**/>**

**</constraints>**

**</view>**

**<connections>**

**<outlet** property**=**"bannerImage"destination**=**"tNd-jS-kdc"id**=**"tvj-fX-jzl"**/>**

**<outlet** property**=**"cityNameLabel"destination**=**"zMn-Ha-b91"id**=**"g8b-fc-f2X"**/>**

**<outlet** property**=**"closeButton"destination**=**"DG4-nx-OKM"id**=**"obL-PM-FXL"**/>**

**<outlet** property**=**"forecastView"destination**=**"IL3-RD-lUu"id**=**"R5R-IM-IHy"**/>**

**<outlet** property**=**"humidityLabel"destination**=**"0y0-Hd-S8d"id**=**"hhO-4Z-heN"**/>**

**<outlet** property**=**"temperatureLabel"destination**=**"UaS-9i-Yyn"id**=**"KGR-IQ-azs"**/>**

**<outlet** property**=**"todayDateLabel"destination**=**"hkH-F7-bAm"id**=**"2eH-VQ-r6E"**/>**

**<outlet** property**=**"todayWeatherDetailsView"destination**=**"x1Q-EE-PMJ"id**=**"3Sk-Sn-WoL"**/>**

**<outlet** property**=**"todayWeatherHeightConstraint"destination**=**"5np-3E-4Du"id**=**"rzN-pW-e1J"**/>**

**<outlet** property**=**"todayWeatherView"destination**=**"0E0-LJ-twe"id**=**"pD6-OM-O3w"**/>**

**<outlet** property**=**"windSpeedLabel"destination**=**"eAg-Yk-ozH"id**=**"ea2-pq-nCg"**/>**

**</connections>**

**</viewController>**

**<placeholder** placeholderIdentifier**=**"IBFirstResponder"id**=**"s2q-Kr-oLq"userLabel**=**"First Responder"customClass**=**"UIResponder"sceneMemberID**=**"firstResponder"**/>**

**</objects>**

**<point** key**=**"canvasLocation"x**=**"956.92307692307691"y**=**"0.7109004739336493"**/>**

**</scene>**

**</scenes>**

**<resources>**

**<image** name**=**"Banner-Image"width**=**"156"height**=**"77"**/>**

**<image** name**=**"Humidity-Image"width**=**"48"height**=**"49"**/>**

**<image** name**=**"SmallCalendar-Image"width**=**"40"height**=**"40"**/>**

**<image** name**=**"Wind-Image"width**=**"48"height**=**"49"**/>**

**<image** name**=**"xmark-Image"width**=**"50"height**=**"50"**/>**

**<namedColor** name**=**"Background-Color"**>**

**<color** red**=**"1"green**=**"1"blue**=**"1"alpha**=**"1"colorSpace**=**"custom"customColorSpace**=**"sRGB"**/>**

**</namedColor>**

**<namedColor** name**=**"SecondaryBackground-Color"**>**

**<color** red**=**"0.11800000071525574"green**=**"0.11800000071525574"blue**=**"0.11800000071525574"alpha**=**"1"colorSpace**=**"custom"customColorSpace**=**"sRGB"**/>**

**</namedColor>**

**<systemColor** name**=**"systemBackgroundColor"**>**

**<color** white**=**"1"alpha**=**"1"colorSpace**=**"custom"customColorSpace**=**"genericGamma22GrayColorSpace"**/>**

**</systemColor>**

**</resources>**

**</document>**

**Файл LaunchScreen.storyboard**

**<?xml version=**"1.0" **encoding=**"UTF-8"**?>**

**<document** type**=**"com.apple.InterfaceBuilder3.CocoaTouch.Storyboard.XIB"version**=**"3.0"toolsVersion**=**"21225"targetRuntime**=**"iOS.CocoaTouch"propertyAccessControl**=**"none"useAutolayout**=**"YES"launchScreen**=**"YES"useTraitCollections**=**"YES"useSafeAreas**=**"YES"colorMatched**=**"YES"initialViewController**=**"01J-lp-oVM"**>**

**<device** id**=**"retina6\_0"orientation**=**"portrait"appearance**=**"light"**/>**

**<dependencies>**

**<deployment** identifier**=**"iOS"**/>**

**<plugIn** identifier**=**"com.apple.InterfaceBuilder.IBCocoaTouchPlugin"version**=**"21207"**/>**

**<capability** name**=**"Named colors"minToolsVersion**=**"9.0"**/>**

**<capability** name**=**"Safe area layout guides"minToolsVersion**=**"9.0"**/>**

**<capability** name**=**"System colors in document resources"minToolsVersion**=**"11.0"**/>**

**<capability** name**=**"documents saved in the Xcode 8 format"minToolsVersion**=**"8.0"**/>**

**</dependencies>**

**<scenes>**

<!--View Controller-->

**<scene** sceneID**=**"EHf-IW-A2E"**>**

**<objects>**

**<viewController** id**=**"01J-lp-oVM"sceneMemberID**=**"viewController"**>**

**<view** key**=**"view"contentMode**=**"scaleToFill"id**=**"Ze5-6b-2t3"**>**

**<rect** key**=**"frame"x**=**"0.0"y**=**"0.0"width**=**"390"height**=**"844"**/>**

**<autoresizingMask** key**=**"autoresizingMask"widthSizable**=**"YES"heightSizable**=**"YES"**/>**

**<subviews>**

**<view** contentMode**=**"scaleToFill"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"Xon-k6-Tqt"**>**

**<rect** key**=**"frame"x**=**"0.0"y**=**"0.0"width**=**"390"height**=**"844"**/>**

**<subviews>**

**<stackView** opaque**=**"NO"contentMode**=**"scaleToFill"axis**=**"vertical"alignment**=**"center"spacing**=**"22"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"DCy-iG-dTJ"**>**

**<rect** key**=**"frame"x**=**"85"y**=**"262.66666666666669"width**=**"220"height**=**"319.00000000000006"**/>**

**<subviews>**

**<imageView** clipsSubviews**=**"YES"userInteractionEnabled**=**"NO"contentMode**=**"scaleAspectFit"horizontalHuggingPriority**=**"251"verticalHuggingPriority**=**"251"image**=**"BigMapPin-Image"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"Mpe-c8-djT"**>**

**<rect** key**=**"frame"x**=**"0.0"y**=**"0.0"width**=**"220"height**=**"260"**/>**

**<constraints>**

**<constraint** firstAttribute**=**"height"constant**=**"260"id**=**"Yrp-dL-4Yx"**/>**

**<constraint** firstAttribute**=**"width"constant**=**"220"id**=**"zJz-xD-eZj"**/>**

**</constraints>**

**</imageView>**

**<label** opaque**=**"NO"userInteractionEnabled**=**"NO"contentMode**=**"left"horizontalHuggingPriority**=**"251"verticalHuggingPriority**=**"251"text**=**"BY #NSString"textAlignment**=**"natural"lineBreakMode**=**"tailTruncation"baselineAdjustment**=**"alignBaselines"adjustsFontSizeToFit**=**"NO"translatesAutoresizingMaskIntoConstraints**=**"NO"id**=**"bX9-yT-ZgP"**>**

**<rect** key**=**"frame"x**=**"11.666666666666671"y**=**"281.99999999999994"width**=**"197"height**=**"37"**/>**

**<fontDescription** key**=**"fontDescription"name**=**"Helvetica"family**=**"Helvetica"pointSize**=**"32"**/>**

**<color** key**=**"textColor"white**=**"1"alpha**=**"1"colorSpace**=**"custom"customColorSpace**=**"genericGamma22GrayColorSpace"**/>**

**<nil** key**=**"highlightedColor"**/>**

**</label>**

**</subviews>**

**</stackView>**

**</subviews>**

**<color** key**=**"backgroundColor"name**=**"DarkGray-Color"**/>**

**<constraints>**

**<constraint** firstItem**=**"DCy-iG-dTJ"firstAttribute**=**"centerY"secondItem**=**"Xon-k6-Tqt"secondAttribute**=**"centerY"id**=**"A6V-Rg-aKj"**/>**

**<constraint** firstItem**=**"DCy-iG-dTJ"firstAttribute**=**"centerX"secondItem**=**"Xon-k6-Tqt"secondAttribute**=**"centerX"id**=**"WtW-KC-rY4"**/>**

**</constraints>**

**</view>**

**</subviews>**

**<viewLayoutGuide** key**=**"safeArea"id**=**"6Tk-OE-BBY"**/>**

**<color** key**=**"backgroundColor"systemColor**=**"systemBackgroundColor"**/>**

**<constraints>**

**<constraint** firstItem**=**"Xon-k6-Tqt"firstAttribute**=**"leading"secondItem**=**"6Tk-OE-BBY"secondAttribute**=**"leading"id**=**"P8Y-T5-SdM"**/>**

**<constraint** firstAttribute**=**"bottom"secondItem**=**"Xon-k6-Tqt"secondAttribute**=**"bottom"id**=**"UJT-C7-cDG"**/>**

**<constraint** firstItem**=**"Xon-k6-Tqt"firstAttribute**=**"trailing"secondItem**=**"6Tk-OE-BBY"secondAttribute**=**"trailing"id**=**"dqX-Qj-JU6"**/>**

**<constraint** firstItem**=**"Xon-k6-Tqt"firstAttribute**=**"top"secondItem**=**"Ze5-6b-2t3"secondAttribute**=**"top"id**=**"iyJ-X6-NNf"**/>**

**</constraints>**

**</view>**

**</viewController>**

**<placeholder** placeholderIdentifier**=**"IBFirstResponder"id**=**"iYj-Kq-Ea1"userLabel**=**"First Responder"sceneMemberID**=**"firstResponder"**/>**

**</objects>**

**<point** key**=**"canvasLocation"x**=**"53"y**=**"375"**/>**

**</scene>**

**</scenes>**

**<resources>**

**<image** name**=**"BigMapPin-Image"width**=**"468"height**=**"539"**/>**

**<namedColor** name**=**"DarkGray-Color"**>**

**<color** red**=**"0.11764705882352941"green**=**"0.11764705882352941"blue**=**"0.11764705882352941"alpha**=**"1"colorSpace**=**"custom"customColorSpace**=**"sRGB"**/>**

**</namedColor>**

**<systemColor** name**=**"systemBackgroundColor"**>**

**<color** white**=**"1"alpha**=**"1"colorSpace**=**"custom"customColorSpace**=**"genericGamma22GrayColorSpace"**/>**

**</systemColor>**

**</resources>**

**</document>**