iOS Labor 5 - Hálózatkezelés

Ez az útmutató végigvezet egy komplett SwiftUI alkalmazás felépítésén ami képes lesz megjeleníteni egy kiválasztott város **aktuális időjárását** és **5 napos előrejelzését**. A felhasználó képes lesz új várost keresni.

Regisztráció és a projekt létrehozása

> Regisztrálj egy ingyenes fiókot az **OpenWeatherMap** weboldalon, és generálj egy személyes **API kulcsot**. Ez elengedhetetlen az adatok lekéréséhez.

> Egy böngésző vagy API kliens segítségével ismerkedj meg a `/weather` (aktuális) és `/forecast` (előrejelzés) végpontok **JSON** válaszával.

Példák:

1. `https://api.openweathermap.org/data/2.5/weather?q=Budapest&units=metric&appid={APP_ID}`
2. `https://api.openweathermap.org/data/2.5/forecast?q=Budapest&units=metric&appid={APP_ID}`

> Figyeld meg a struktúrát, a kulcsneveket és az adattípusokat.

> Hozz létre egy új, **SwiftUI** alapú **iOS App** projektet az Xcode-ban.

Adatmodellek Felépítése

> Hozz létre egy új Swift fájlt, pl. `WeatherModels.swift`. Ebben a fájlban lesznek definiálva a kérésekre kapott válaszokat leíró struct-ok.

> Az elemzett JSON struktúra alapján hozd létre `struct`-okat.
> Készíts egy-egy modellt
>
> - Az aktuális időjárásra, ami tartalmazza a város `nevét`, a `hőmérsékletet`, a `szelet` és az `időjárás leírását`.
> - Az előrejelzésre, ami egy listát tartalmaz a jövőbeli adatpontokból.

> Minden létrehozott `struct` feleljen meg a **`Codable`** protokollnak. Ez teszi lehetővé, hogy a `JSONDecoder` automatikusan átalakítsa a hálózati választ a te Swift modelljeiddé.

Egy lehetséges megoldás:

```
struct CurrentWeatherData: Codable {
    let name: String
    let main: Main
    let weather: [Weather]
    let wind: Wind
}

struct Main: Codable {
    let temp: Double
    let feels_like: Double
```

```
    let humidity: Int
}

struct Weather: Codable {
    let description: String
    let icon: String
}

struct Wind: Codable {
    let speed: Double
}

struct ForecastData: Codable {
    let list: [ListItem]
}

struct ListItem: Codable, Identifiable {
    var id: Int { dt }
    let dt: Int
    let dt_txt: String
    let main: Main
    let weather: [Weather]
}
```

Hálózati Kommunikáció

> Hozz létre egy `WeatherService.swift` fájlt benne egy `WeatherService` nevű `class`-szal.

> Tedd az osztályodat `Observable`-lé az `@Observable` makróval, hogy a SwiftUI nézetek figyelni tudják a benne történő változásokat. Ne felejts el egy `import`-tal hivatkozni a `SwiftUI` framework-re.

> Hozz létre property-ket a letöltött adatok: **aktuális időjárás**, **előrejelzés**, a **betöltési állapot** (`isLoading`) és az **esetleges hibaüzenetek** (`errorMessage`) tárolására.

> Írj egy-egy `async` függvényt a két funkcióhoz, amik:
>
> - Biztonságosan felépítik a teljes URL-t a városnév és az API kulcsod segítségével.
> - Az `URLSession` segítségével, `async/await` használatával letöltik az adatokat.
> - A `JSONDecoder` segítségével a kapott JSON-t a korábban létrehozott **`Codable`** modelljeiddé alakítják.
> - A sikeresen dekódolt adatokat a megfelelő **`@Published`** változókba mentik.
> - A teljes folyamatot `do-catch` blokkba ágyazzák a hibakezelés érdekében.

Egy lehetséges kiindulási alap:

```
func fetchWeather(for city: String)async {
    guard let url = URL(string:
"https://api.openweathermap.org/data/2.5/weather?q=\(city)&appid=\
(apiKey)&units=metric")
```

```
        else {
            return
        }

        guard let response = try? await URLSession.shared.data(for:
    URLRequest(url:url))
        else {
            return
        }

        do {
            self.weatherInfo = try JSONDecoder().decode(WeatherInfo.self,
    from: response.0)
        }
        catch(let error) {
            print("Error decoding: \(error)")
        }
    }
```

Főképernyő

> Hozz létre egy `MainWeatherView.swift` nevű SwiftUI View fájlt.

> Hozz létre egy példányt a `WeatherService`-ből **`@State`** property wrapperrel.

> Használj `VStack`, `HStack` és `Spacer` elemeket a dizájn kialakításához.

> Helyezz el `Text` elemeket az adatok (város, hőmérséklet, stb).

> Az időjárás ikon megjelenítéséhez használj `AsyncImage` nézetet.

Példa az `AsyncImage` használatára:

```
        AsyncImage(url: URL(string: "https://openweathermap.org/img/wn/\
    (iconName ?? "01d")@2x.png"))
```

> A nézet logikája kezelje a `WeatherService` állapotait:
>
> - Ha az `isLoading` igaz, mutasson egy `ProgressView`-t.
> - Ha az `errorMessage` nem üres, jelenítse meg a hibát.
> - Ha az adatok rendelkezésre állnak, mutassa az időjárási információkat.

> Használd a `.task` módosítót a nézeten, hogy automatikusan elindítsd az adatletöltést a képernyő megjelenésekor.

Keresés

Tegyük interaktívvá az alkalmazást.

> Hozz létre egy új `CitySearchView.swift` SwiftUI View fájlt.

> Adj hozzá egy `TextField`-et a városnév beírásához és egy `Button`-t a keresés elküldéséhez.

Figyelj oda, hogy a Button action-jében az async hívást egy `Task{...}`-on belül valósítsd meg.

> Használj `@Binding`-ot, hogy a kereső nézet frissíteni tudja a fő nézetben tárolt, keresett város nevét.

```
@Binding var currentCity: String
```

> A fő nézetből egy gombnyomásra jelenítsd meg a kereső nézetet egy `.sheet` módosítóval (felugró lapként).

Előrejelzés

Hozzunk létre egy dedikált felületet a részletes adatoknak.

> Hozz létre egy új `ForecastView.swift` SwiftUI View fájlt, aminek a fő nézet adja majd át a letöltött előrejelzési adatokat, amikor a felhasználó odanavigál.

> Használj `List`-et az előrejelzési adatok soronkénti megjelenítéséhez. Minden sor tartalmazza a dátumot, a hőmérsékletet és az ikont.

> Mivel az API 3 órás adatokat ad, írj egy logikát (pl. egy `computed property`-t), ami leegyszerűsíti a listát, és naponta csak egy (pl. a déli) előrejelzést mutat.

Valahogy így:

```
let forecast: ForecastData

private var dailyForecasts: [ListItem] {
    forecast.list.filter { $0.dt_txt.contains("12:00:00") }
}
```

Végső Összeállítás és Navigáció

Fésüld össze az elemeket egy működő egésszé.

> A `MainWeatherView`-t csomagold be egy `NavigationStack`-be, hogy a `NavigationLink` működjön, és a felhasználó átjuthasson az előrejelzés képernyőjére.

```
NavigationStack{
    ...
    if let forecast = weatherService.forecastData {
        NavigationLink("5 napos előrejelzés") {
            ForecastView(forecast: forecast)
        }.buttonStyle(.borderedProminent)
    }
}
```

> Az alkalmazás központi `App` protocolt megvalósító `@main` fájljában állítsd be, hogy a `MainWeatherView` legyen az induló nézet.

> Futtasd az alkalmazást. Próbálj ki érvényes és érvénytelen városneveket, és ellenőrizd, hogy a betöltési állapot, az adatmegjelenítés, a hibakezelés és a navigáció is a tervek szerint működik-e.