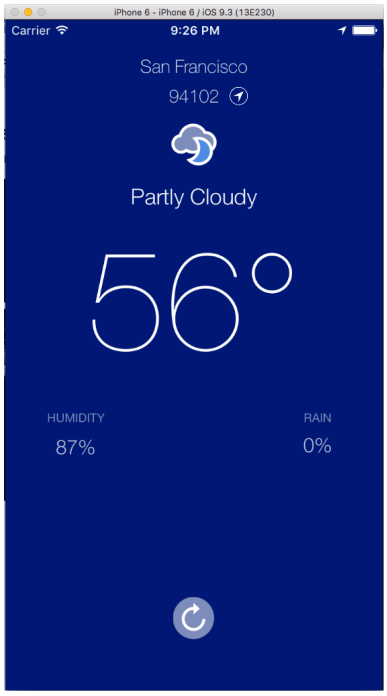
CS496_400_U2016
Matt Mayberry
07/31/2016

Assignment 4

Video Site URL: https://limitless-ocean-45014.herokuapp.com/

1.0 Overview:

For this assignment I created an iOS weather app, called 'Forecast' written in Swift and utilizing the Forecast.io API. The app submits GET requests to the service and populates the UI to show the user weather data about their current location.



2.0 Web Service:

The Forecast API (https://developer.forecast.io/docs/v2) allows developers to get weather information for specified locations. The client sends GET requests to the API and the Forecast API returns JSON-formatted object with the following properties defined:

- `latitude`: The requested latitude.
- `longitude`: The requested longitude.
- `timezone`: The IANA timezone name for the requested location (e.g. `America/New_York`). This is the timezone used for text forecast summaries and for determining the exact start time of daily data points.*(Developers are advised to rely on local system settings rather than*

this value if at all possible: users may deliberately set an unusual timezone, and furthermore are likely to know what they actually want better than our timezone database does.)

- `offset`: The current timezone offset in hours from GMT. *(This value is deprecated and should not be used.)*
- `currently`: A *data point* (see below) containing the current weather conditions at the requested location.
- `minutely`: A *data block* (see below) containing the weather conditions minute-by-minute for the next hour.
- `hourly`: A *data block* (see below) containing the weather conditions hour-by-hour for the next two days.
- `daily`: A *data block* (see below) containing the weather conditions day-by-day for the next week.
- `alerts`: An array of *alert objects* (see below), which, if present, contains any severe weather alerts, issued by a governmental weather authority, pertinent to the requested location.
- `flags`: An object containing miscellaneous metadata concerning this request.

Source: https://developer.forecast.io/docs/v2

3.0 Making Requests

The session is initiated by fetchWeatherResults, within the ForecastCommunicator.swift file. After which the request is created in the ForecastRequest.swift, outlined below.

ForecastRequest.swift

```
import Foundation
import CoreLocation

struct ForecastRequest {
    let cLat: CLLocationDegrees
    let cLon: CLLocationDegrees

    var stringLatLon: String {
        return "/\(cLat),\(cLon)"
    }

    var baseURLString: String {
        return "https://api.forecast.io/forecast/" + "e7d8d21f3e7c1c515d68fba89aa058ba"
    }

    var requestURLString: String {
        return baseURLString + stringLatLon
    }

    var requestURL: NSURL? {
        return NSURL(string: requestURLString)
    }
```

4.0 Using the returned JSON object

The returned JSON-object is then stored in a dictionary and the dictionary is used to populate the variables within the UI, and the appropriate weather icon is selected in the switch case.

```
struct CurrentWeatherModel {

    var currentTime: String?
    var temperature: Double
    var humidity: Double
    var precipProbability: Double
    var summary: String
    var icon: UIImage?

    init(weatherDictionary: NSDictionary) {
        let currentWeather = weatherDictionary["currently"] as! NSDictionary

        let currentTimeIntValue = currentWeather["time"] as! Int

        temperature = currentWeather["temperature"] as! Double
        humidity = currentWeather["humidity"] as! Double
        precipProbability = currentWeather["precipProbability"] as! Double
        summary = currentWeather["summary"] as! String
        currentTime = dateStringFromUnixTime(currentTimeIntValue)

        let iconString = currentWeather["icon"] as! String
        icon = weatherIconFromString(iconString)
    }


    func dateStringFromUnixTime(unixTime: Int) -> String {
        let timeInSeconds = NSTimeInterval(unixTime)
        let weatherDate = NSDate(timeIntervalSince1970: timeInSeconds)

        let dateFormatter = NSDateFormatter()
        dateFormatter.timeStyle = .ShortStyle

        return dateFormatter.stringFromDate(weatherDate)
    }


    func weatherIconFromString(stringIcon: String) -> UIImage {
        var imageName: String

        switch stringIcon {
        case "clear-day":
            imageName = "clear-day"
        case "clear-night":
            imageName = "clear-night"
        case "rain":
            imageName = "rain"
        case "snow":
            imageName = "snow"
        case "sleet":
            imageName = "sleet"
        case "wind":
            imageName = "wind"
        case "fog":
            imageName = "fog"
        case "cloudy":
            imageName = "cloudy"
        case "partly-cloudy-day":
            imageName = "partly-cloudy"
        case "partly-cloudy-night":
            imageName = "cloudy-night"
        default:
            imageName = "default"
        }

        let iconImage = UIImage(named: imageName)
        return iconImage!
    }
```

5.0 CoreLocation Framework

The Core Location frameworks lets an app get the current position of the user's device. In the case of the Forecast app, I'm using this information to add the user's current lat and long to the GET request for the purpose of obtaining the weather for the users current location.

```
struct ForecastRequest {
    let cLat: CLLocationDegrees
    let cLon: CLLocationDegrees

    var stringLatLon: String {
        return "/\(cLat),\(cLon)"
    }
```