iOS-App

```
11  struct Sensor {
12      var tempValues     : [Data] = [];
13      var humidValues    : [Data] = [];
14      var windValues     : [Data] = [];
15      var pressureValues : [Data] = [];
16
17      struct SensorData : Codable {
18          public class SensorDataValue : Codable {
19              let time     : String
20              let value    : Double
21              let dataType : String
22          }
23
24          let temp     : SensorDataValue
25          let humidity : SensorDataValue
26          let pressure : SensorDataValue
27          let wind     : SensorDataValue
28      }
29
30      struct Data : Identifiable {
31          let id         = UUID()
32          let date       : Date
33          let value      : Double
34      }
35
36      enum LocalError : Error {
37          case runtimeError(String)
38      }
```

Aplicația de iOS are la bază structura Sensor, care conține SensorData, obiectul care este citit din URL. Aceasta are deasemena structuri ajutătoare pentru puncte de date și erori.

```
40      public init() async throws {
41          let data = try! await getSensorHistory()
42          let dateFormatter = DateFormatter()
43          dateFormatter.dateFormat = "yyyy'-'MM'-'dd'T'HH':'mm':'ss'Z'"
44
45          for sensorData in data {
46              let toAppend = Data(
47                  date: dateFormatter.date(from: sensorData.time).unsafelyUnwrapped,
48                  value: sensorData.value
49              )
50              switch sensorData.dataType{
51              case "temp":
52                  self.tempValues.append(toAppend)
53              case "humidity":
54                  self.humidValues.append(toAppend)
55              case "wind":
56                  self.windValues.append(toAppend)
57              case "pressure":
58                  self.pressureValues.append(toAppend)
59              default:
60                  throw LocalError.runtimeError("Wrong data in JSON :(")
61              }
62          }
63      }
```

Constructorul structurii Sensor realizează citirea datelor de la API și îi introduce în vectorii de date

```swift
    public func getSensorHistory() async throws -> [SensorData.SensorDataValue] {
        guard let url: URL = URL(string: "https://awu4j6hku3.execute-api.eu-central-1.amazonaws.com/dev/weather/hist?days=1&group=24") else {
            throw LocalError.runtimeError("URL GET Failed :(")
        }

        do {
            let (data, _) = try await URLSession.shared.data(from: url)

            let decoded: [SensorData.SensorDataValue] = try JSONDecoder().decode([SensorData.SensorDataValue].self, from: data)

            return decoded
        } catch {
            throw LocalError.runtimeError("Failed to decode URL to JSON Array :(")
        }
    }

    public func getSensorData() async throws -> SensorData {
        guard let url: URL = URL(string: "https://awu4j6hku3.execute-api.eu-central-1.amazonaws.com/dev/weather/latest") else {
            throw LocalError.runtimeError("URL GET Failed :(")
        }

        do {
            let (data, _) = try await URLSession.shared.data(from: url)

            let decoded: SensorData = try JSONDecoder().decode(SensorData.self, from: data)

            return decoded
        } catch {
            throw LocalError.runtimeError("Failed to decode URL to JSON :(")
        }
    }
```

functiile getSensorHistory() și getSensorData() ambele fac un request http GET la API, doar că o realizează la un URL diferit, și returnează tipuri de date diferite
cele două funcții declară un URL, din care ulterior extrag date, care sunt decodate ca fișier json.

```swift
    public func getTemperatureValues() -> [Data] {
        return tempValues
    }

    public func getHumidityValues() -> [Data] {
        return humidValues
    }

    public func getWindValues() -> [Data] {
        return windValues
    }

    public func getPressureValues() -> [Data] {
        return pressureValues
    }
```

cele 4 funcții wrapper care returnează cele 4 valori de senzori

```swift
struct LineChartView : View {
    let data      : [Sensor.Data]
    let chartName : String

    var body : some View {
        VStack {
            GroupBox {
                Text(chartName)
                    .bold()
                Chart {
                    ForEach(data) {
                        LineMark(
                            x: .value("Time", $0.date, unit: .hour),
                            y: .value(chartName, $0.value)
                        )
                    }
                }
            }
        }
    }
}
```

structura de tip View pentru realizarea chart-ului cu linii, pentru care este declarat un titlu, și sunt plasate punctele pe grafic

```swift
public struct ContentView : View {
    @State private var temperatureValues : [Sensor.Data] = []
    @State private var humidityValues    : [Sensor.Data] = []
    @State private var windValues        : [Sensor.Data] = []
    @State private var pressureValues    : [Sensor.Data] = []

    public var body: some View {
        VStack(alignment: .leading) {
//            Text("Temperature").bold()
//                .foregroundColor(.red)
//
//            BarChartView(data: temperatureValues, colors: [.red, .orange])
            LineChartView(data: temperatureValues, chartName: "Temperature")
                .foregroundColor(.red)

//            Text("Humidity").bold()
//                .foregroundColor(.blue)
//
//            BarChartView(data: humidityValues, colors: [.blue, .purple])
            LineChartView(data: humidityValues, chartName: "Humidity")
                .foregroundColor(.blue)

//            Text("Wind").bold()
//                .foregroundColor(.green)
//
//            BarChartView(data: windValues, colors: [.green, .yellow])
            LineChartView(data: windValues, chartName: "Wind")
                .foregroundColor(.green)

            LineChartView(data: pressureValues, chartName: "Pressure")
                .foregroundColor(.yellow)
        }
        .padding()
        .task {
            let Sensors = try! await Sensor()
            temperatureValues = Sensors.getTemperatureValues()
            humidityValues = Sensors.getHumidityValues()
            windValues = Sensors.getWindValues()
            pressureValues = Sensors.getPressureValues()
        }
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

View-ul principal, care realizează cele 4 grafice după ce realizează task-ul de a declara obiectull de Sensor și asigenază vectorii pentru valorile de puncte de grafic