Ozan Topal

Central Weather Control

Software Design Document

Date: 23/09/2021

Table of Contents

Definitions and Acronyms

Abbreviation	Description
SDD	Software Design Document
UI	User Interface
CLR	Common Language Runtime
AMQP	Advanced Message Queuing Protocol
STOMP	Streaming Text Oriented Messaging Protocol
MQTT	MQ Telemetry Transport
REST	REpresentational State Transfer
JSON	JavaScript Object Notation
MQ	Message Queue
UUID	Universally Unique Identifier
PoC	Proof of Concept
KISS	Keep It Simple, Stupid
YAGNI	You Aren't Gonna Need It
POCO	Plain Old CLR Object
UML	Unified Modelling Language

1 INTRODUCTION

1.1 Purpose

This document is intended to provide information about the architectural software designs. Also, this document keeps the main role as information exchange object between stakeholders.

Main purpose of this document is specifying how to fulfill the requirements. Therefore, this document contains design language selections, viewpoint design documentations.

1.2 Scope

Weather might be one of the most changing situations in the globe and also it affects most of our lives in so many different ways. We can list that ways such as agriculture, transportation and traffic and so on.

Beside this, measurement of weather situation can vary slightly by affecting by geographical shapes around. For eliminating that change caused by environmental reason, should position weather station around that geographical shapes and them take their position in calculation. However, locating so many conventional weather stations is going to create cost for building and operating it. Instead of that fully installed weather stations this project is aiming to design central data process and management unit which is communicating with all local measurement devices.

1.3 Overview

This SDD is divided into four main parts. Those parts are explained below:

- System Overview gives you a general description of the functionality, context and design of the project.
- System Architecture gives you information about system's architectural design, and also this part includes documentation diagrams.
- Data Design part is about to define and describe data generated and transferred by the system.
- Human Interface Design will inform you about the functionality from the user's perspective. This part also contains mock UI screens for simulating experience end users will have.

2 SYSTEM OVERVIEW

Central Weather Control system is mainly parted to four uniquely responsibility component. These are measurement unit, message queue, data collector and central result unit.

Project can be extended by different components such as end user information systems which can work on web, desktop and also mobile environment too. Also, there can be different units like neural network and etc. but in the scope of this project we just going to focus on four main component we mention above.

Measurement units are our local devices that listen periodically its own sensors to create weather information where it's located. Functionality of this component is basically reading

data from its sensors and enqueue them into message queue of the system. The data produced by measurement unit will be explained in detail in section data design.

Message queue is such a data bridge between measurement units and message consumer. Its function is to keep the system alive by preventing the data collector from becoming inoperable. This component is also an open-source message-broker software named as RabbitMQ. It's written in Erlang and support several protocols like AMQP, STOMP, MQTT and others. For further information about RabbitMQ please visit its homepage and GitHub profile.

Data collector is a job-based application. It's basically subscriber of message queue. It's responsible for dequeuing messages one by one and then saving to the data source.

Central result unit is RESTful endpoint of the system that querying and calculating (in case of need) data and present in a specific format. This format is mostly preferred as JSON.

3 SYSTEM ARCHITECTURE

3.1 Architectural Design

Within the scope of the project, many different architectural designs could be considered, but I developed my PoC code in the simplest way I could, adhering to the KISS and YAGNI principles. That's why I prefer to develop three executable projects. Those are; MeasurementUnitEmulator, DataCollector, CentralResultUnit. I also add a common project for POCOs.

There's also sequence and UML diagrams below:

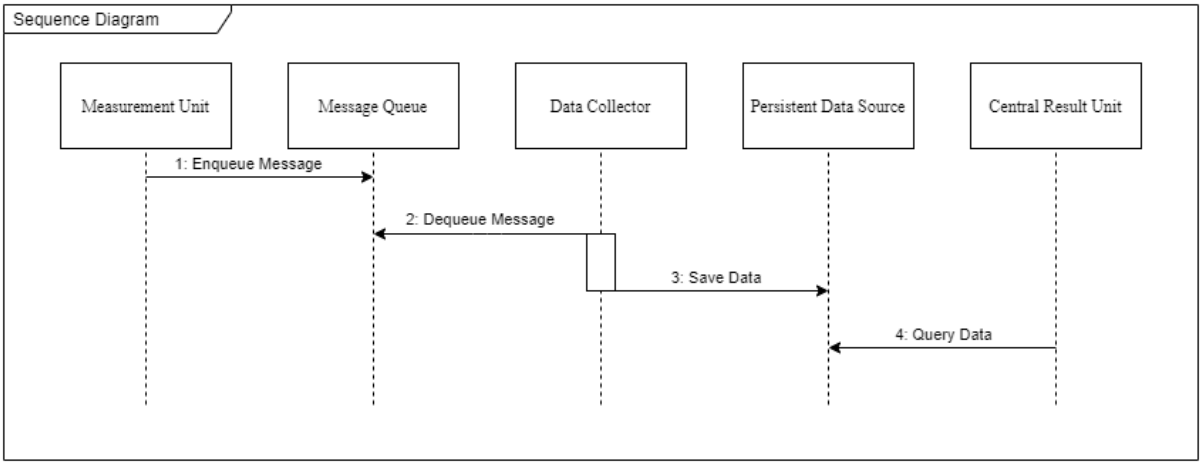

Figure 1: Sequence Diagram of system

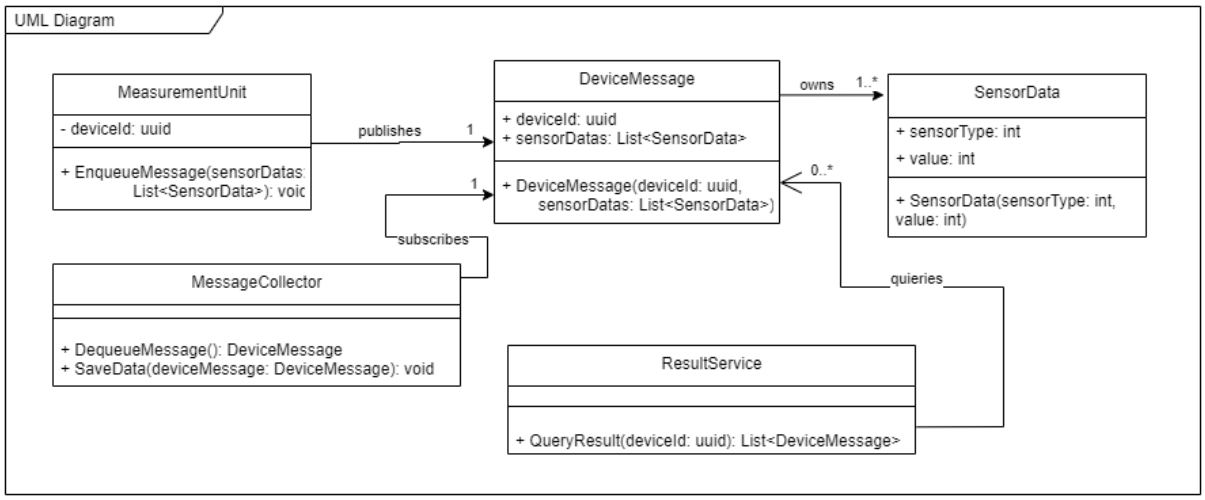

Figure 2: UML Diagram of system

3.2 Design Rationale

3.2.1 Design Decisions

DD-1: It was decided to use third-party MQ instead of developing it in-house. Main reason for taking that decision is developing a message broker has its own challenge like handle develop cost and also test and maintain cost too. On the other hand, RabbitMQ is well-known message broker for consistency.

4 DATA DESIGN

4.1 Data Description

DeviceMessage which will be enqueued by measurement unit basically contains two fields. Those are deviceId which is a uuid and sensorDatas which is list of SensorData.

Also, SensorData object has two fields and one of them is about to describing type of sensor, other one keeps data read from sensor. Type of sensor, named as sensorType, is an enumeration. Values of that enumeration listed below:

- 1 for temperature
- 2 for humidity
- 3 for wind speed

Our imaginary weather control sensors make these three different measurements in the following ranges and sensitivities:

Temperature can have value between -20 and +40 and unit of that value is °C. Humidity is the amount of water vapor in the air. This field can take values from 0 to 100 and unit of that value is percent. Finally, wind speed takes values between 0 and 50 and this value is in type of knot. Knot should also be mentioned here. It’s fundamental unit of measurement for wind speed and equals nautical mile per hour. It also equals 1.852 km/h.

5 HUMAN INTERFACE DESIGN

5.1 Overview of User Interface and Screen Images

Graphical user interface is excluded from this task but a basic proposal mockup is inserted for giving an idea.

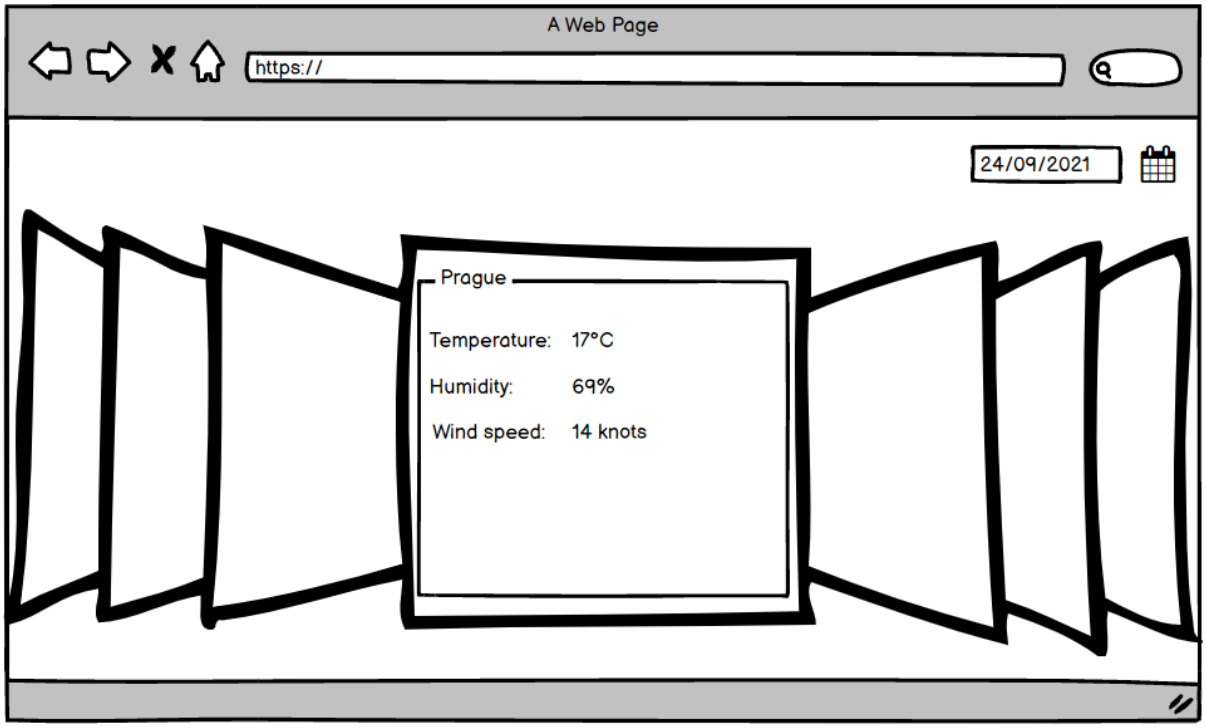

Figure 3: Proposal mock user interface