The recommended architecture to analyse IoT Data for asset management

Bachelor Thesis

Submitted by Bader Ammoun

In fulfillment of the requirements for the degree Bachelor of Science in Informatics To be awarded by The Fontys Hogeschool Techniek en Logistiek

Sittard July 6, 2020

Fontys University of Applied Sciences
School of Technology and Logistics
Post Office Box 141, 5900 AC Venlo, Netherlands

Type of report: research

Student name: Bader Ammoun
Student number: 3437310
Study: Software Engineering
Period: 10-2-2020 To 10-7-2020

Company name: BCT
Address: Hub Dassenplein 3
Postal code + City: 6130 AB Sittard
Country: Netherlands
Telephone: +31(0)46 442 45 45

Company supervisor: Math.Huntjes
Company supervisor: Laurens.van der Blom
Supervising Lecturer : Frank.Gennip
External commissioner: Th.Dorssers

Company Confidential: Yes

Number of words: 9990

STATEMENT OF AUTHENTICITY

Issued by the FHTenL Examination Board, September 2017

I, the undersigned, hereby certify that I have compiled and written this document and the underlying work / pieces of work without assistance from anyone except the specifically assigned academic supervisor. This work is solely my own, and I am solely responsible for the content, organization, and making of this document and the underlying work / pieces of work.

I hereby acknowledge that I have read the instructions for preparation and submission of documents / pieces of work provided by my course / my academic institution, and I understand that this document and the underlying pieces of work will not be accepted for evaluation or for the award of academic credits if it is determined that they have not been prepared in compliance with those instructions and this statement of authenticity.

I further certify that I did not commit plagiarism, did neither take over nor paraphrase (digital or printed, translated or original) material (e.g. ideas, data, pieces of text, figures, diagrams, tables, recordings, videos, code, ...) produced by others without correct and complete citation and correct and complete reference of the source(s). I understand that this document and the underlying work / pieces of work will not be accepted for evaluation or for the award of academic credits if it is determined that they embody plagiarism.

Name: Bader Ammoun

Student number: 3437310

Place/Date: Sittard July 6, 2020

Signature: _____

Contents

List of Figures

List of Tables

1 Summary

This report investigates the architecture of a system that provides solution to the challenges of analyzing an enormous amount of IoT data. The investigation is based on research, articles, and books, which deal with various relevant aspects, in addition, to the comparisons between options and the selection of the most appropriate one. Further it sets up the testing framework to verify if the recommended architecture fulfills the requirements by determining what needs to be tested and how to conduct tests. Finally, it evaluates the results and what can be done in the future.

2 Introduction

This chapter starts with discussing what this report is intended for. Then it moves on giving a short introduction about the company for which the project has been working on, in addition to the problem that the company tries to solve. Then it concludes with a review of the subsequent parts of this report that were discussed as part of tackling the problem to reach to the solution.

2.1 Background

This thesis is intended to obtain a bachelor's degree in software engineering from Fontys university of applied sciences

2.2 Company description

BCT is a family company, founded in 1985, and currently has a settlement in Sittard. Through innovation, high quality and excellent services BCT has grown into a company with at least 170 employees and a revenue of 13 million Euro. BCT has a strong position in the Dutch (semi-)government market. BCT's customers are knowledge intensive organizations who require accuracy, completeness, reliability and availability of information. BCT guides customers with the implementation of integral information management. BCT has successfully assisted at least 700 customers to make their ambitions in information transition come true. BCT analyses, advises and offers the correct software solutions. The customer's organization is always the heart of an implementation by BCT, not the software systems involved. This is due to the fact that every organization is different and information management is always specific to the organization.

2.3 Problem Statement

Thanks to the IoT technology, it has become convenient to observe and convey the physical changes in the assets to a digital platform. The digital platform can track changes with these assets and evaluate the current situation depending on the current data. Meanwhile, keeping the historical data is useful for business

intelligence reports meaning, it may reveal hidden pattern and correlations between different environmental factors that can influence the asset. Accordingly, investing in historical data has become essential for most companies that invest in the management asset market to stay in the competition. In contrast, the failure to take advantage of this data inevitably leads to the company losing its market. However, investing these enormous amounts of data encompass many challenges. These challenges concern mainly on transferring, processing, storing, and finally extracting the beneficial information to observe the trends.

2.4 Project objectives

The main objective of this project is to conduct research about what is a recommended software architecture that can adders challenges that are stated in the problem statement.

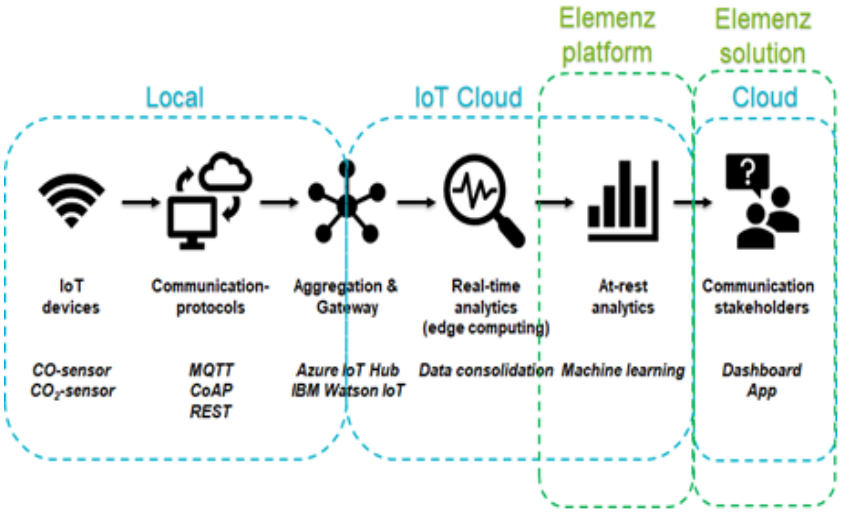

Figure 1: project objective

2.5 Air quality use case

Addressing these challenges needs to work on a concrete use case. The air quality use case uses many IoT devices to produce a large amount of data which, makes it a good choice. The air quality around and within buildings and structures is known as indoor air quality. Poor indoor air quality can irritate the employee's eyes, nose, and throat, or can result in fatigue, nausea, or illness. The health effects of these symptoms can affect the employee's well-being and lead

to poor work performance and productivity. On the other hand, the employer is responsible for ensuring a safe and healthy work environment. To keep the indoor air healthy, it is vital to know the level of pollution factors and how they change over time. The aim of the air quality project in the high-level overview is to distribute a group of sensors throughout the employee's rooms to measure the real-time level for temperature, and humidity, then Storing data in the management system before consolidated it in a warehouse for analysis and visualization. This process helps the facility manager to find out different trends then take appropriate action depending on the data.

2.6 Overview

Here is a list of chapters and a summary of what has been discussed in each them

- Analysis and Requirement Describes how the requirements were elicitation, And arrived at the specified requirements.
- Development Process Describes the development process that was followed to achieve the stated objectives.
- Architecture describes the architectural design model with the following strategy. It is reviewing the literature, articles, books in different areas that are related to the context of this project. It is Making a comparison of different approaches then, Choose which one is fit according to the requirements of the project.
- Threats modeling reviews the potential risks and how to mitigate them.
- Prototype discusses the assumptions that would simplify some of the solutions to demonstrate the concept of the architecture.
- Implantation discuss the reasons why the specific programming language, development tools, and the implementation platform were chosen.
- Test and Validation Describe the test strategy and how to conduct the test to fulfill the requirements
- Conclusion the problem was summarized besides to what has been achieved, and prospect for future work.

3 Analysis and Requirements

This chapter explains briefly how the functional requirements have been gathered, and Then it moves to review them alongside to the user characteristics and non-functional requirements

3.1 Methods

The methods that were used to elicit the requirements are arranging interviews with the facility manager and using questionnaires with multi choices as feedback to ensure that the requirements were understood.

3.2 Use characteristics

The facility manager who wants to maintain healthy air quality and act properly in a timely manner for any problems that may occur in this context.

3.3 Functional Requirement

Name	Visualize the historical data
ID	UC1
Priority	Must have
Description	The Facility manager wants to visualize the historical data to have insight that helps him to arrange an appropriate action for a particular case.

Table 1: UC1 User Case

Name	Air Conditioner Maintenance's Notification
ID	UC2
Priority	May have
Description	The purpose of Air Conditioner Maintenance Notification is to start using conditional maintenance instead of scheduled maintenance, which in it turns reduces the costs.

Table 2: UC2 User Case

3.4 Use Case Diagram

Figure 2: Use case Diagram

3.5 Non-Functional Requirements

Following is talking about the non-functional requirements, Describing the essential attributes in this system.

Performance	
ID	NF1
Priority	Must have
Purposes	IoT device produces a significant amount of data. fetching them and processing them should be done with high performance.

Table 3: Performance Requirement

Availability	
ID	NF2
Priority	Must have
Purposes	The IoT devices produce the data continuously. Thus, the system should be functional all day.

Table 4: Availability Requirement

Security	
ID	NF3
Priority	Must have
Purposes	The system should allow only an authorized user to get access to the dashboard. And secure data between sensors and the system.

Table 5: Security Requirement

Scalability	
ID	NF4
Priority	Must have
Purposes	Many IoT devices connect to the system. adding more devices in the future is possible thus, adding the device to the system should not influence the performance.

Table 6: scalability Requirement

4 Development Process

The iterative framework has been followed as the development process. The sec-

IterateNo	Deliverable
1	Visualizing historical data
2	Air conditioner maintenance notification

Table 7: Iterative plan

ond iterate will be implemented as part of the project's future. The first iterate is the most important one, and by achieving it, all important and fundamental challenges will be be addressed.

5 Architecture

This chapter provides a comprehensive architectural overview of the system, using a number of different architectural views to depict various aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system. The architecture should meet all functional non-functional requirements. However, scalability and availability are essential pillars to the quality of the system; thus, the solution should consist of many subsystems, and every subsystem should be built as discrete services that are independently deployable, and able to scale independently. These attributes enable greater scale, more flexibility in updating individual subsystems, and provide the flexibility to choose appropriate technology on a per subsystem basis. Additionally, those subsystems should support fault tolerance principle in case one service is down for some reason there is another instance of this service that can take its role and let the whole system continue to operate. How the system‘s architecture will achieve scalability, and availability will be illustrated in the upcoming sections. However, regarding the security, there is a chapter discussing it besides, to the test and validation issueRichardson (2019). The chapter starts with review the logical view of the system, and then it moves to provide an overview of the internal implantation for every micros-service and how they communicate.

5.1 Logical views(Micro-services)

This subsection explains every service individually in very high level abstraction.It discuss the reasons for its presence in the context of functional and non-functional requirements.

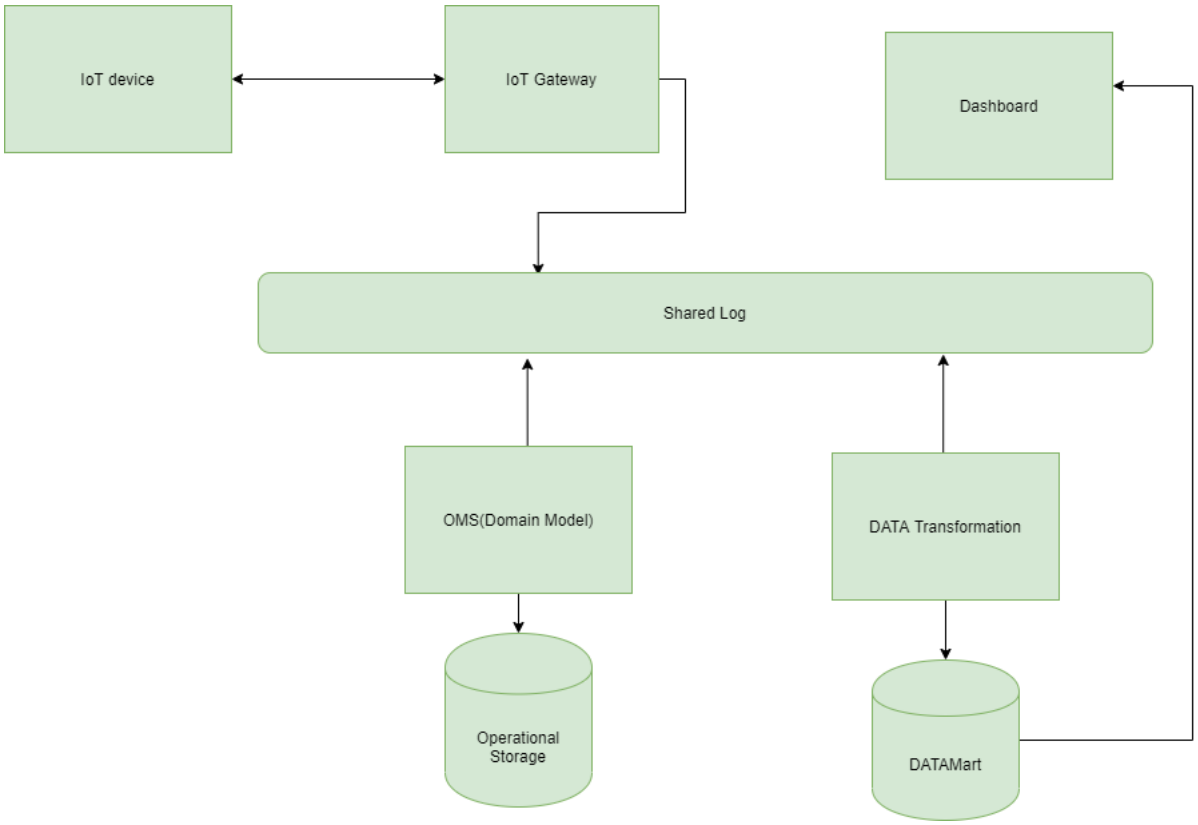

Figure 3: Reference Architecture

5.1.1 Warehouse(Data Mart)

Let's take a closer look at the UC1 case: visualizing the historical data. this could encompass the following:

1. The facility manager wants to find out the trends of fact(Temperature, Humidity, etc.) in one room during a specific time window.
2. The facility manager wants to know the total value for fact on the floor or building.

As we can see, there are unlimited possibilities for the reports that the facility manager wants. And all of them are essentially the same, but the difference is in slicing and dicing the values depending on the time, floor, and the nature of the fact(temperature, humidity). Apart from that, let's take a look at the definition of Business intelligence (BI). According to Marky Lk the definition (RADACAT-Team 2016) "leverages software and services to transform data into actionable insights that inform an organization's business decisions ". Which, exactly what

we have discussed previously and according to him "the Transactional databases built for CRUD operations (Create, Retrieve, Update, Delete rows). Because of this single purpose, transactional databases are built in a Normalized way, to reduce redundancy and increase the consistency of the data, In fact, building a data model for BI systems needs to be avoided. This model works perfectly for transactional databases (when there are systems and operators do data entry and modifications). However, this model is not good for a BI system. There are several reasons for that, here are two most important reasons; The model is hard to understand for a Report User. Too many tables and many relationships between tables make a reporting query (that might use 20 of these tables at once) very slow and not efficient" . Accordingly, there is a need for a different type of Transactional database, which is called a data warehouse. There are several types of schema("the discussion about them takes place later on in this section"). But, apart from that, what is worth mentioning in this regard is, the needing for the operational database is still vital as long as there is live transaction coming back and forth to the system. Here the notion of data transformation emerges, which is nothing more than the process of converting operational data into one of the warehouse schemes that are suitable for the business intelligence report. Before diving into the type schemes discussion, It is important to grasp the two important concepts of the warehouse world, and their responsibilities (RADACAT-Team 2016)

- "A Fact table is a table that keeps numeric data that might be aggregated in the reporting visualizations".
- "A Dimension table is a table that keeps descriptive information that can slice and dice the data of the fact table.

now let's see the type of schemes

- Star schema : Central table whose primary key is compound, i.e., consisting of multiple attributes. Each one of these attributes is a foreign key to one of the remaining tables. Such a foreign key dependency exists for each one of these tables, while there are no other foreign keys anywhere in the schema. (In the above, without loss of generality, the assumption is made that all these other tables have simple primary keys. This is usually the case in almost all practical situations, as for efficiency, these keys are typically generated surrogate keys.)A star schema has one "central" table whose primary key is compound, i.e., consisting of multiple attributes. Each one of these attributes is a foreign key to one of the remaining tables. Such a foreign key dependency exists for each one of these tables, while there are no other foreign keys anywhere in the schema. (In the above, without loss of generality, the assumption is made that all these other tables have simple primary keys. This is usually the case in almost all practical situations, as for efficiency, these keys are typically generated surrogate keys.Chaudhuri & Dayal (1997)
- Snowflake : The snowflake schema is a variant of the star schema. Here, the centralized fact table is connected to multiple dimensions. In the

snowflake schema, dimensions are present in a normalized form in multiple related tables. The snowflake structure materializes when the dimensions of a star schema are detailed and highly structured, having several levels of relationship, and the child tables have multiple parent tables. The snowflake effect affects only the dimension tables and does not affect the fact tables.Chaudhuri & Dayal (1997)

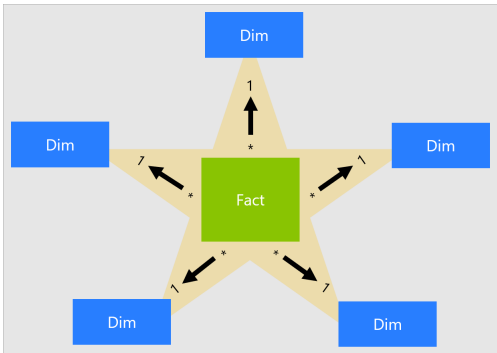

Figure 4: star-schema-example Myers et al. (2019)

In conclusion, In any business intelligence project, there are Operational database to perform crud operations and A warehouse which is nothing else than the relational database but with a different schema. When modeling this schema, it should be borne in mind that the purpose is to slice and dice the data depending on many descriptive properties. Later on, This scheme will be used by Business intelligence report tools. The implementation view section discusses the implementation of the air quality's star schema.

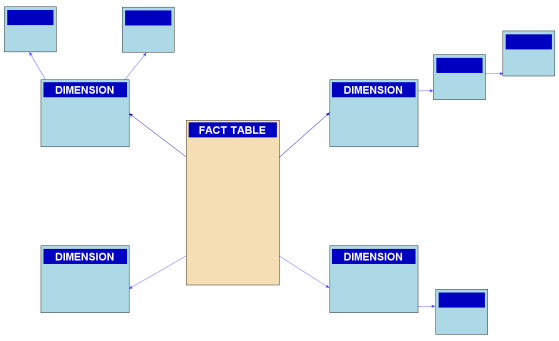

Figure 5: Snowflake-schema-example Hernandez (2018)

5.1.2 Dashboard

It is a component of every BI software solution. The main task is to allow users to receive instant visualization of their preferred BI-specific operations, eliminating requirements for manually executed queries or processes. Moreover, a BI dashboard's appearance and interface may be customized for desktop, mobile, or Web/cloud users. Building a dashboard for business intelligence from scratch is cumbersome, costly, and most likely, the outcome will be not flexible for the business user's needs, especially there are many tools available in the market. Reviewing and comparing them is out of the scope of this thesis. However, the power BI desktop from Microsoft has been chosen as a visual tool because it works in grate compatibility with the warehouse database. All what it needs is the scheme that has been chosen in the warehouse section.

5.1.3 Data transformation

Data transformation can increase the efficiency of analytic and business processes and enable better data-driven decision-making. However, the data transformation concerns how to convert operational data to an analysis model(star or snow flow schema). Most likely, the converting process encompasses the following Filtering, aggregation, and summarization.

5.1.4 OMS (Object Management system)

It stands for Object Management System. It's the implementation of a system that allows modeling any real-world object (either physical or abstract) as a digital object. That makes it possible to gather data about these objects, such as buildings, rooms, and so on, in our context, then check if the buildings compile with the governance roles in a different aspect. Surely this data is persisted in the operation database.

5.1.5 Cloud gateway

The sensors are mentioned in the last paragraph. However, getting information securely from sensors and managing them is a tough, not easy job. So heading to the cloud is the best option in this regard. According to Microsoft, the cloud gateway "is A cloud gateway that enables remote communication to and from devices or edge devices, which potentially reside at several different sites. A cloud gateway will either be reachable over the public Internet, or a network virtualization overlay (VPN), or private network connections into Azure datacenters, to insulate the cloud gateway and all of its attached devices or edge devices from other network traffic. It generally manages all aspects of communication, including transport-protocol-level connection management, protection of the communication path, device authentication, and authorization toward the system. It enforces connection and throughput quotas and collects data used for billing, diagnostics, and other monitoring tasks. The data flow from the device

through the cloud gateway is executed through one or multiple application-level messaging Protocols.microsoft (2018)

5.1.6 IoT device

It interacts with the physical world; it senses physical parameters, which in our case are (temperature, Humidity) and sends it securely to the IoT cloud gateway. The following discusses the type of network and the protocols in the IoT world Setting up the IoT network can be divided into two distinct parts:

- Part concerns with the physical and data link layer.
- Part concerns with the application layer.

physical and data link layer

1. Cellular: This kind of network is distributed through areas called "cells". One fixed-location transceiver serves at least one cell. The cell uses the transceiver to transmit voice, data, and other types of content. Usually, the cell uses different frequency form its neighbor to prevent the interference.Dunko et al. (2017)

Advantages	Disadvantages
• Connect anywhere, anytime. • Low power. • Penetrate solid barriers. • Secure. • Ip-driven connection	• Cellular carriers infrastructure is costly • Cellular carriers needs a specific skills and knowledge. In most cases, depending on the third party to operate and maintain the network, is the best choice.

Table 8: Cellular comparison Dunko et al. (2017)

2. WiFi: WiFi is capable of connecting to the network with high speed and without wires. It uses radio frequencies to send data between devices. It bases on the IEEE 802.11 family of standards, which are used for local area networking of devices and the Internet access.Dunko et al. (2017)

Advantages	Disadvantages
• It does not need recurring cost. • low cost. • No bandwidth restriction. • Low latency Than Cellular. • Ip-driven connection. • Maintenance and operation of network are not costly.	• Space limitation. • More Power consumption. • Does not penetrate solid barriers • Less secure than cellular. • The connections between devices and central data center are fully dependent on the router's connection to the Internet.

Table 9: WiFi comparison Dunko et al. (2017)

3. Bluetooth: Bluetooth exchanges data between devices within a short distance. it uses short-wavelength radio waves from 2.400 to 2.485 GHz and building personal area networks (PANs).Dunko et al. (2017)

Advantages	Disadvantages
• Low power consumption. • Inexpensive.	• Space limitation. • None Ip driven connection. • Interference with other device. • Low security.

Table 10: Bluetooth comparison Dunko et al. (2017)

4. Zigbee ZigBee is an excellent choice for creating personal area networks with small, low-power digital radios. These networks, such as for home automation, medical device data collection, and other low-power low-bandwidth needs, designed for small scale projects which need wireless connection. Zigbee is IEEE 802.15.4-based specification .Dunko et al. (2017)

Advantages	Disadvantages
• Low power consumption. • Inexpensive.	• Space limitation. • None Ip driven connection. • Low bandwidth. • Low security.

Table 11: Zigbee comparison Dunko et al. (2017)

The following clarify the correlation between speed and distance in the different type of IoT network

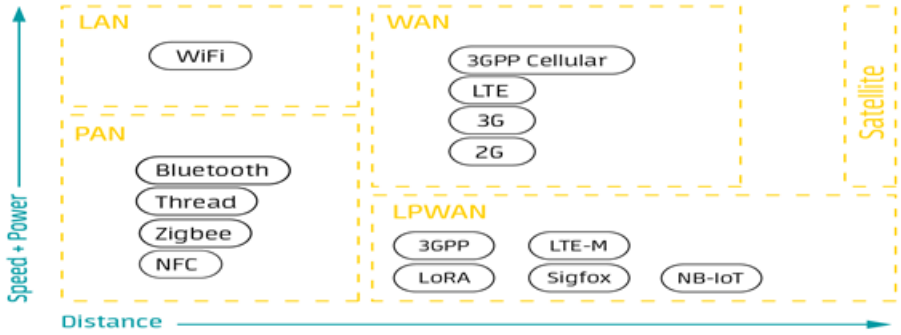

Figure 6: Network Connectivity Dunko et al. (2017)

Protocol Application

1. Mqtt: MQTT is one of the most commonly used protocols in IoT projects. It stands for Message Queuing Telemetry Transport. In addition, it is designed as a lightweight messaging protocol that uses publish/subscribe operations to exchange data between clients and the server. Furthermore, its small size, low power usage, minimized data packets and ease of implementation make the protocol ideal for the "machine-to-machine" or "Internet of Things" world.Sethi & Smruti (2017)

Advantages	Disadvantages
• It's a lightweight protocol. So, it's easy to implement in software and fast in data transmission.. • Low power usage. As a result, it saves the connected device's battery. • It's real time! That's specifically what makes it perfect for IoT applications.	• MQTT provides no support for labelling messages with types or other metadata to help clients understand it..

Table 12: Mqtt comparison Sethi & Smruti (2017)

2. CoAp:CoAp is a Internet Application Protocol for constrained devices. It allows those constrained devices to communicate with the larger node. CoAP is designed for devices to consume less power and send data to general node on the internet.Sethi & Smruti (2017)
   The main differences between CoAP and Maqtt are.
   - The first aspect to notice is the different paradigm used. MQTT uses a publisher-subscriber while CoAP uses a request-response paradigm.
   - MQTT uses a central broker to dispatch messages coming from the publisher to the clients. CoAP is essentially a one-to-one protocol very similar to the HTTP protocol.
   - Moreover, MQTT is an event-oriented protocol while CoAP is more suitable for state transfer.
3. HTTP HTTP is not suitable in resource constrained environments because
   - Slow: because it uses bigger data packets to communicate with the server.

- Overhead: HTTP request opens and closes the connection at each request.
- Power consuming: since it takes a longer time and more data packets, therefore it uses much power.Sethi & Smruti (2017)

5.2 Implementation view

Before going into details, it should be noted that the c4 model has been adopted to be used as a visualization tools to depict the system architecture for many reasons:

- It has the high descriptive ability by showing a 4 level overview, starting with a high-level overview of the system then it goes deeper and deeper
- It can capture the static and dynamic parts of the system.Brown (2019)
- Architects of the BCT use it to document the systems, and besides, they use it as an illustration tool in the meetings and their blueprint.

The c2 level shows the system at a high-level overview, including all the subsystems and the connection protocols among them. Moreover, the figure shows some services that were not mentioned in architecture reference but are essential to some subsystems for performing their tasks.

5.2.1 Subsystems structure

This section explains the implementation of each micro-service individually, and How does it work internally. The powerful BI desktop will be used as a business intelligence report tool, and The Azure cloud platform will be used as a cloud gateway. The implementation chapter justifies the reasons for choosing them. However, this section focus is on how the data will be extracted, brought, consolidated, and transferred to the star model.

1. IoT device
   Every room has its instance, and it senses the (temperature, Humidity) on a minute basis and sends its value to the gateway, Only if it detects a change in value from the last measurement, the repeated data is avoided to be sent. In the subsection logical view, the different types of IoT networks and IoT protocol applications were reviewed; thus, let's choose what the most appropriate choices for the project are.
   - WiFi
     The project will be implemented inside the building, which leads to the exclusion of the cellular option. On the other hand, Being the Zigbee does not IP-driven network puts it off the list; thus, The perfect fit is wifi, Especially the project is inside a building.

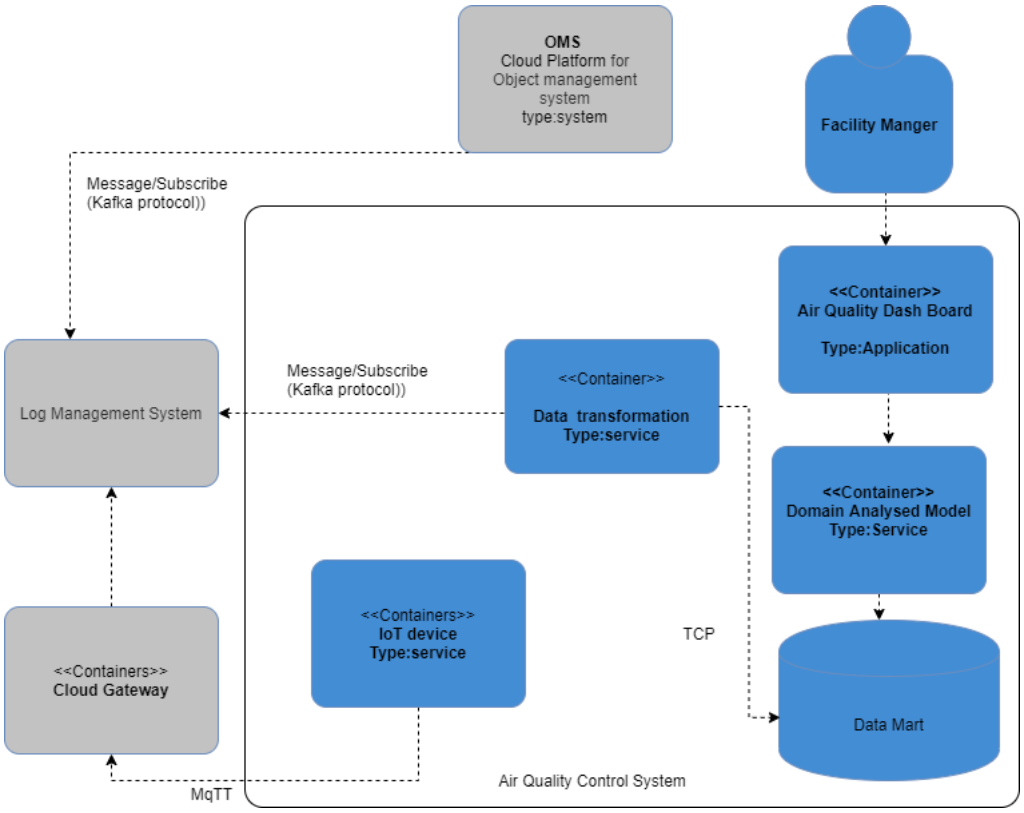

Figure 7: Container Diagram

- MQTT
  MQTT and Coap both of them are perfect for any IoT project. But the coap is stateless application protocol like HTTP. Consequently, the data transfer process will be somewhat static, and the connection is one-directional. All the factors may affect the system and its effectiveness in real-time. Therefore the MQTT is a good choice.

2. OMS
   Although OMS is a part of the reference architecture, Its design and implementation are being carried out by BCT. However, since it is in the development phases and not available yet through the internship, the chapter proof of concept reviews its internal implementation according to the author's perspective.

3. Data Log Management System
   Before goes into details about the need for log, let's find out the two concepts: state mutation and mutable event by concrete example related to the context of the project. IoT device produces data consistently. But not

all services in the system interest in all this data, for example, some services like OMS is interested only with the last current record or maybe on other records to perform its business logic. Once the IoT device produces new facts, the OMS will update the mapped record on its database. Another service like data transformation is interested only on the aggregation of values within a specific time window, as we will see later. In conclusion, all rows of data need to be stored somewhere, and later any service can fetch the data that it is interested in. This raw data is a mutable event, whereas, the OMS uses a state mutation to update the record. Coming back to storing the raw data. Surely, storing it in relation database could be the simplest solution, but it is so limited in terms of scalability. For more clarification, let assume that there are two instances of data transformation service that consume data from a relational database in this scenario; maintaining the consistency of data without repeating the same data in two services becomes a nightmare. Alternatively, store all records in a fixed order, and apply them in that fixed order to the various places they need to go. Whenever any IoT device writes a new data, this data will be appended to the end of a sequence of records. That sequence is totally ordered, it's append-only (never modify existing records, only add new records at the end), and it's persistent (we store it durably on disk). Structure the data in this way simplifies the consistency of the data in different services in the system. In conclusion, both log and relation database is similar in terms of purpose, which is persisting the data. But both of them use a different structure to store data. This structure will determine later how to query the data. Using a log provides an opportunity for any service in the system to read and write data at a frequency that suits its business logic. Any changes in requirements, whether on the service level or on the system level like adding new features, do not need any changes in the architecture meaning, the architecture is extendable, not modifiable.Kleppmann (2016) Now that the notion of the log has been discussed beside the reasons for using it, it is time to review the mechanism of Fetching data from log to the data transformation service:

(a) Batch:"A batch is a collection of data points that have been grouped together within a specific time interval. Another term often used for this is a window of data(Balkenende 2018)

(b) Stream: "Streaming processing deals with continuous data and is key to turning big data into fast data".(Balkenende 2018)

Batch Advantages	Batch Disadvantages
• Batch Processing is a good choice for processing large volumes of data/transaction. • The processing of data can be done independently. at a desired designated time. • carrying out the process using batches brings to the company the cost efficiency. • good audit trail.	• The delay between the collection of data and getting the result after the batch process. • In the batch processing the data is out of date. • one-time process can be very slow.

Table 13: Batch comparison (Balkenende 2018)

Streaming Advantages	Streaming Disadvantages
• carrying out the real-time processing brings instantly response. • In real-time processing, information is always up to date. • By using streaming the organization gains insights from the data and detect the hidden patterns by machine without humane interference.	• Real-Time processing is very complex and expensive processing.

Table 14: Steaming comparison (Balkenende 2018)

It is time to put this discussion in the project context and choose which method is the best fit for the requirements. Whether Applying the batch or streaming mechanisms does not add any extra functions to any microservice in the system. All the complexities are managed by the log management system. These complexities encompass, delivering data to the interested service at the desired frequency and ensure the consistency of the data. In conclusion, it can be said that the streaming can do what the batch does, but the ver versa is not correct, on the other hand, the second requirement(UC2) needs to be done in real-time thus, the streaming option is the best fit for the project requirements.

4. Data warehouse
   In the logical views section, the warehouse was indicated now, let's see how the model that meets the user requirements is built. first, let's start with the dimension tables The graphs that facility manager wants to see could be sliced and diced depending on the room (temperature, humidity), floor, and time; thus, they are dimension tables. Since the (temperature, Humidity) could vary depending on the usage of the room(meeting, working, storing) and the number of the people who can present during the work hours, adding these attributes to the room dimension table enriches the business intelligence report. Currently, there are no more descriptive attributes that could be added to the floor except the number of the floor, likewise to the building except the address. However, adding more attributes related to theses two entities if the business requirements change is more flexible thanks to treating them as a dimension table. On the other hand, due to the fact of the correlations between (temperature, humidity) and seasons of the year, adding the season to the time dimension allows gaining more insights about the air quality inside the building. Likewise, there is a correlations between these facts and the part of the day(morning, noon, evening, night); thus, the part of the day is an attribute in the time dimension. Now let's move to the fact table. The facility manager aims to find out the trends of every air quality facts in the room, floor, and building. Since IoT sends facts in minutes basis while the interest is only on the parts of the day, there is a need to calculate the average value for every part of day alongside, to the max and min for every individual part.

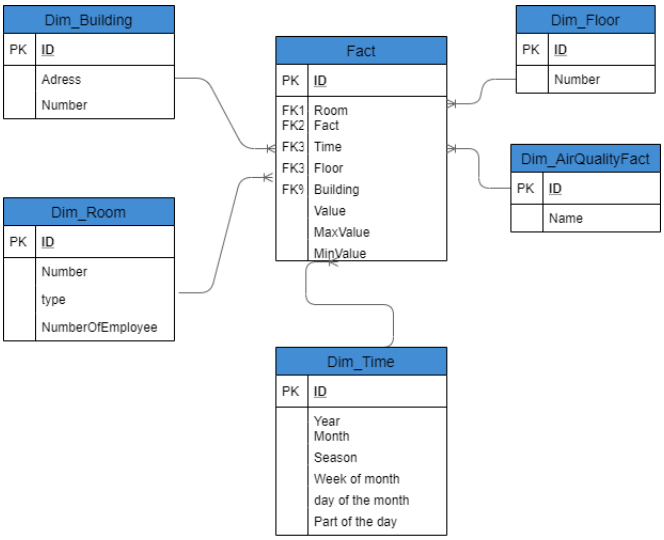

Figure 8: star schema

5. Data Transformation
   The converting from the operational data schema to the analytical data schema is the main task of this microservice. Figure 18 in the prototype chapter states the schema of the operational database which needs to be converted to the star schema model as figure 8 states. In the data log management system subsection, the decision has been taken to stream the events to this service, so First, let's discuss how this service process the stream then moves to address other components on this service.

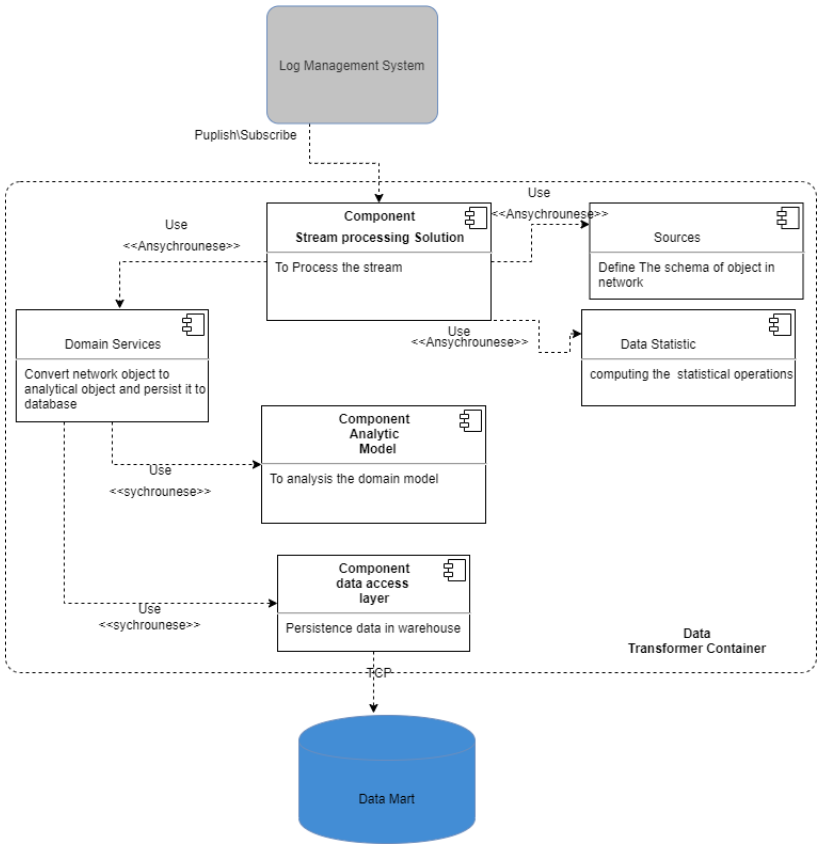

Figure 9: Data Transformation Component

(a) Stream Processing
topology determines how input data is transformed into output data. Any topology consists of a graphs of stream processors (nodes) that are connected by streams (edges) or shared state stores. let's highlight every individual term in this definition.

Stream processor A stream processor is a node in the topology. where operations such as filtering,joining, and aggregation are preformed. It receives one input record at a time from its upstream processors(node) in the topology, applies its operation to it, and may subsequently produce one or more output records to its downstream processors.(cofluent 2019) There are two special processors in the topology:

- Source Processor: A source processor is a special type of stream processor that does not have any upstream processors. It produces an input stream to its topology from the data log by consuming records from it and forward them to its down-stream processors.
- Sink Processor: A sink processor is a special type of stream processor that does not have down-stream processors. It sends any received records from its up-stream processors to a specific data storage".(cofluent 2019)

Stream A stream represents an unbounded, continuously updating data set.

State store state store is used in Stateful stream. statefull means that a "state" is shared between events and therefore past events can influence the way current events are processed".(Narkhede et al. 2017) whereas,"In a Stateless stream, the way each event is handled is completely independent from the preceding events. Given an event, the stream processor will treat it exactly the same way every time, no matter what data arrived beforehand".(Narkhede et al. 2017) there are three type of store

- key-value store the Stream can be considered a changelog of a table, where each data record has the same key updates the value of the same key in the table.
- windowing store gives the capability to define a fix time window then group the records depending on the key and the window. In this way, when a new record arrived to the stream, there are two possibilities, either this record will be added to the table under a new key or aggregated its value with the record that has the same key in the table then update the value of the key in the table.

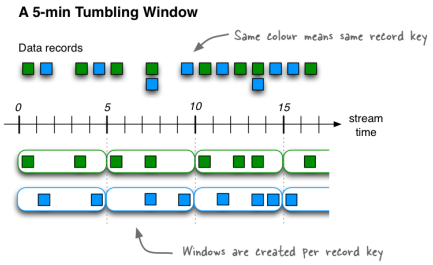

Figure 10: Time window (cofluent 2019)

- "Session windows are used to aggregate key-based events into so-called sessions, the process of which is referred to as sessionization. Sessions represent a period of activity separated by a defined gap of inactivity (or "idleness"). Any events processed that fall within the inactivity gap of any existing sessions are merged into the existing sessions. If an event falls outside of the session gap, then a new session will be created.Session windows are different from the other window types in that:all windows are tracked independently across keys – e.g., windows of different keys typically have different start and end times their window sizes sizes vary – even windows for the same key typically have different sizes".(cofluent 2019)

After defining the terms and the concepts of streaming processing , let's put those terms in the project context and see how the requirements can be met. OMS service uses Kafka producer API to stream events to the Data Analysis service by using validate state topic. Every room state has its key, which is a room number. Having the same key for every room state event ensures that those events will be transmitted to the same broker, and the same consumer will consume it. Considering that many consumers could consume the records, this is important to ensure the consistency of the room's statistics, which requires to maintain the states of past events locally. As previously discussed in the data warehouse section, one of the requirements is to calculate the average value of the facts environments on a sex hours basis. Accordingly, the stream processing is statefull. we have discussed the topology of the stream processing and its elements thus, let's start with the

i. Stream processors: The source is the log, whereas the sink is the data mart. Group by key, group by time window, and aggregation are nodes in between the source and sink, and they convert the data to the desired data.

ii. group by groups the events that belong to the same room.

iii. Group by time window takes the outcome of the previous node and groups them in such a way that the events that belong to the same time window will be the outcome of this node.

iv. The aggregate node uses a state time window store to maintains the reference to the previous event, then add the current value to it then, update the store with the new value.

v. The store keeps updating as long as new events come to the stream; thus, the stream needs to emit the value once the time window is close to the downstream.

vi. The downstream catches the value and persist it to the data mart.

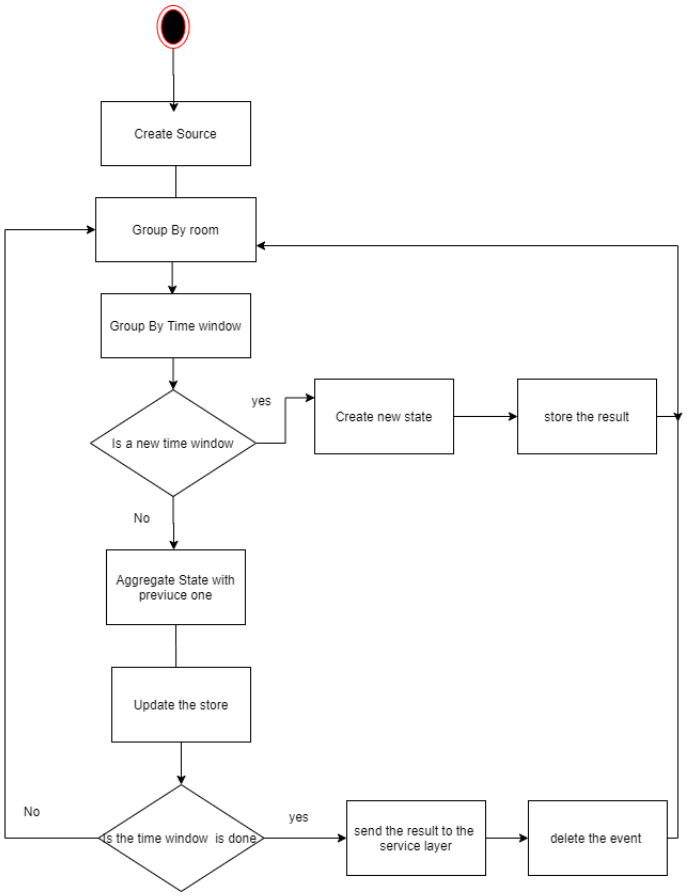

Figure 11: Data flow diagram

(b) Statistic Component

This component encapsulates all the logic of statistical operations. It

takes the current event form stream and returns the analytical results to the stream again.

(c) Analytic model
The Analytic model maps the model of the data mart.

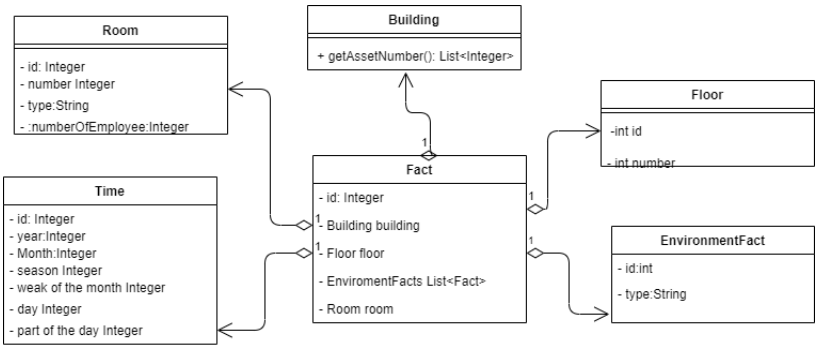

Figure 12: Analytic model class diagram

For the sake of making this subsection concise and concentrating on what is matters to the main task of data transformation microservice, the explaining of source and domain service components takes place at the prototype chapter. There are two components in the OMS that follow the same principle, which is domain-driven design. The OMS subsection at prototype chapter discusses this notion with more details.

5.2.2 Communications between subsystem

After reviewing all microservices, it's time to see how these services communicate and exchange the data. We have seen that the scalability and fault tolerance are the motives behind choosing the shared log. Accordingly, let's briefly review the meaning of these notions in the context of a distributed system. A distributed system consists of many independent components running in a different machine. Those components interact with each other through a network.Burns (2018) The fault-tolerance notion indicates the ability of the system to continue to operate despite the failure of one or more of its components Surely, the failure of all the system's components is elusive to be coped but, it could be said that more failures can be tolerated, the higher is the resilience to failures and the dependability of the distributed system in general ".(Storm 2011) Scalability is a very crucial factor in a distributed system. It refers to the ability of the system to increase its performance by increasing the physical resources dynamically .(Network 2018) Achieving Scalability falls into two ways. Scale by increasing the physical resources(RAM, CPU) or scale by adding more machines into the pool of resources. The first one, called vertical Scalability, whereas the second one called horizontal Scalability. Traditional consuming data form shared log fall into two categories: Shared Message Queues and Publish-Subscribe models.

- Shared Message Queue
  In A shared message queue the system makes the messages available in a queue. Thus, once the consumer gets a message at a time, the message will be deleted from the queue meaning, each message pushed to the queue is read by one consumer. Consumer pull the message from the end of the queue that being shared amongst them.(Goswami 2018) Accordingly, this model can not fulfill the scalability and fault tolerance.

- Publish-Subscribe Systems In this model, Many publishers send messages to topics hosed by brokers; meanwhile, multiple subscribers subscribe to a specific topic, and each one of them gets all messages from that topic. Scalability is limited as each subscriber must subscribe to every partition to access the messages from all partitions. Thus, while traditional pub-sub models work for small networks, the instability increases with the growth in nodes".(Goswami 2018)

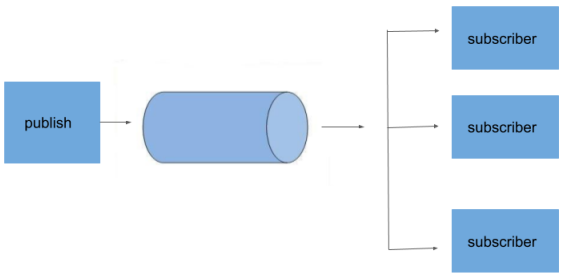

Figure 13: publish-subscriber (Goswami 2018)

- Kafka
  Kafka follows the publish-subscribe model but with slightly different. The notion of group consumer and message retention are the reasons for that difference. Group consumers make Kafka take the advantages of both message queuing and publish-subscribe models. Kafka consumers that belong to the same consumer group share the same id. Consuming from the topic is fairly distributed among all the consumers in the consumer group. As a consumer group scales up and down, the running consumers split the partitions up amongst themselves. Rebalancing is triggered by a shift in ownership between a partition and consumer which could be caused by the crash of a consumer or broker or the addition of a topic or partition. It allows for safe addition or removal of the consumer from the system. When the consumer startup, it requests metadata from the Kafka cluster. The metadata contains the list of the topics, the number of partitions, and the leader of each partition to start request data. In the other hand, the leader of the partition traces the consumer state to detect the consumer failure by listening to the heartbeat from consumer once

the consumer fails to send the heartbeat during a specific time period the leader of the partition marks the consumer as dead and rebalancing the work among the live consumers in the group.(Goswami 2018).Accordingly, "Kafka's flexible scalability makes it easy to handle any amount of data. Users can start with a single broker as a proof of concept, expand to a small development cluster of three brokers, and move into production with a larger cluster of tens or even hundreds of brokers that grows over time as the data scales up. Expansions can be performed while the cluster is online, with no impact on the availability of the system as a whole. This also means that a cluster of multiple brokers can handle the failure of an individual broker and continue servicing clients. Clusters that need to tolerate more simultaneous failures can be configured with higher replication factors".Narkhede et al. (2017)

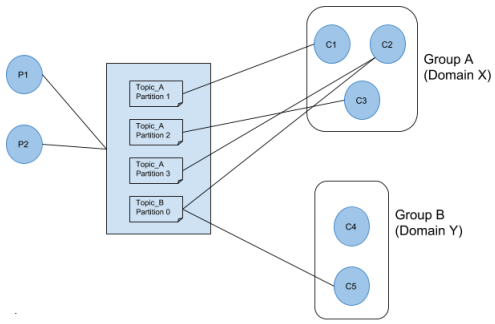

Figure 14: Kafka-Consumer-Groups (Goswami 2018)

The following explains the millstone concepts of Kafka.Kleppmann et al. (2017)

1. Kafka broker
   In high-level overview its responsibilities are:
   - To receive a message from the producer and acknowledge the successful receipt.
   - Store the messages in a log file to safeguard it from potential loss.
   - Deliver the messages to the consumers when they request it.
2. Topic
   Is a logical name to group the message. When a producer sends messages to the cluster over a topic then only those messages will be consumed by the consumer who subscribes to this topic.
3. Producer
   Produce the messages and send them to clusters over a topic.
4. Consumer
   Consume the messages that are coming from a topic.

5. Partition
   This is very important in terms of scalability. The topic can be divided into parts. Each part can be hold by a single broker. This broker can be allocated to one consumer in the group of consumers. by default Kafka use the key to determine the Partition of the message in the topic.

6. Record
   Record has key and message. the message could be a simple plain text or number or even complex object whereas the key is used to determine which partition will receive the record. the records that has the same key will be received by the same broker.

Embedding Kafka API in all microservices makes the scalability of these services trivial since all the complexity will move to the Kafka. All we need is to set the configuration in the right way. But here we must pay attention to a very important issue which is the data transformation service use a local store and consequently the correction of data will be compromised when a decision will be made to scale up this service. The following example demonstrates why: assuming there are two instances of data transformation service and one message belongs to the room one has been processed by the first data transformation instance and it has been stored at a local store now, the second message they belong to the same room and time window has arrived and the second instance of the data transformation service has started to processed the message since the second instance does not have access to the local store to the first instance the result will be wrong. To prevent such a scenario, each message from the same room has one key, which is a room number. Having the same key for every message belongs to the same room ensures that those messages will be transmitted to the same broker, and the same consumer will consume them. The general picture of the mechanism that is used to communicate between microservices is following: Any microservice can write data to the topic in the shared log. Meanwhile, any service can read data from that topic, does its job and, rewrite the data to the new topic. Cloud gateway pushes the telemetry data to the topic called IoT. OMS reads the data does its business logic and rewrite the outcome to the Analysis topic in the shared data log. The data transformation reads the data form analysis topic, then it aggregates, transforms, persists the data in the data warehouse, and finally writes the result in a new topic for any service that can be added in the future, and it needs this data. In the second development iteration, this service is(Machine learning service). Dashboard reads the data from the data warehouse by creating its internal domain model. The sequence diagram illustrates all these steps.

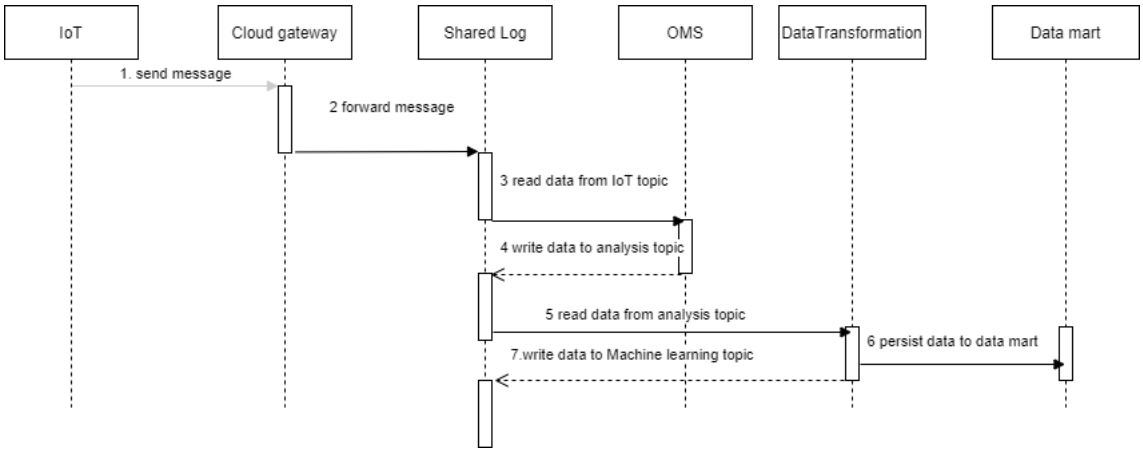

Figure 15: Sequence Diagram

6 Threats modeling

This section aims to understand how an attacker might be able to compromise a system and then make sure appropriate mitigation is in place. The model considers mitigation as the system is designed rather than after a system is deployed. This fact is critically important because retrofitting security defenses to a myriad of devices in the field is infeasible, error-prone, and leaves customers at risk.

6.1 Threat type

According to Microsoft , the types of threats can be classified as follows: Shahan et al. (2018)

- Spoofing: Spoofing in IT world indicates to the deceive the system. usually the attacker tries to hide his identity or to falsifying it.
- Tampering: It refers to an attempt to modify data in a harmful way, usually throughout the unauthorized channel. For instance when data is sent over a wire , It is more likely to be modified maliciously by the intruder and consequently, undermine the system.
- Repudiation: This term refers to the lack of proof that someone made an illegal attempt in the system. The reason for the lack of proof is due to the lack of the system's ability to trace and prohibit the operation. Non-Repudiation refers to the ability of a system to counter repudiation threats. For example, a user who purchases an item might have to sign for the item upon receipt. The vendor can then use the signed receipt as evidence that the user did receive the package.
- Information Disclosure: Involves the exposure of information to the third party who does not have right to access it. For instance the user can read a file that he does not have permissions to access , or the ability of an intruder to read data in transit between two computers
- Denial of Service:Denial of service (DoS) The attacker managed successfully to prohibit the service from the valid-user example, by making a Web server temporarily unavailable or unusable. The system should have the ability to handle certain types of DoS by improving the availability and reliability of the system.
- Elevation of Privilege: happens when Unauthorized user gets the privilege to access the system Consequently, the system treats him as a trusted system giving him the opportunity to destroy the entire system

6.2 Potential threats and mitigation

Referring to the design architecture there are four different regions of the Potential threats. Mainly between:IoT device and cloud gateway, (cloud gateway,

OMS, Data transformation) and shared log. The connection between cloud gateway,OMS,data Transformation and shared log are essentially same, thus the threats and mitigation are similar. The process of modeling threats is composed of four steps:

- Identify the region of the threats.
- Enumerate threats.
- Prioritize threat. The priority is asses depending on the likelihood of the threat occurrence and the impacts in case it occurs.
- Mitigate threats.

6.2.1 Threats between cloud and IoT device

Type	Spoof
Likelihood	Very likely
Impact	Compromising the correction of data
Priority	High
Description	An adversary may replace the IoT Device or part of the IoT Device with some other IoT Device.
Mitigation	Ensure that devices connecting to Field or Cloud gateway are authenticated.

Table 15: Region-1- spoof-threat

Type	Information Disclosure
Likelihood	Very likely
Impact	disclosure the data to illegal party
Priority	High
Description	An adversary may eavesdrop and interfere with the communication between IoT Device and IoT Cloud Gateway and possibly tamper the data that is transmitted.
Mitigation	Secure Device to Cloud Gateway communication using SSL/TLS.

Table 16: Region-1- Disclosure-threat

Type	Repudiation
Likelihood	less likely
Impact	Inability to block the party who preform unauthorized action temporally or permanent period
Priority	Medium
Description	There may be spoofing attempts of devices, unauthorized access to the cloud gateway and so on, all of which must be proven so that deniability of such events or actions is impossible.
Mitigation	Ensure that appropriate auditing and logging is enforced on Cloud Gateway .

Table 17: Region-1- Repudiation-threat

Type	Elevation of Privileges
Likelihood	less likely
Impact	Compressing the correction of data
Priority	Medium
Description	An adversary may leverage insufficient authorization checks on the device and execute unauthorized and sensitive commands remotely.
Mitigation	Perform authorization checks in the device if it supports various actions that require different permission levels.

Table 18: Region-1- Elevation of Privileges-threat

6.2.2 Threats between Cloud,OMS,Data Transformation

Type	Tampering
Likelihood	Very likely
Impact	Compromising the correction of data
Priority	High
Description	An adversary may inject malicious inputs into the log and affect on stream
Mitigation	Ensure that only trusted service can read and write data to the shared data log. .

Table 19: Region-2- Tampering-threat

Type	Spoofing
Likelihood	Very likely
Impact	Compromising the correction of data
Priority	High
Description	If proper authentication is not in place, an adversary can spoof a source process or external entity and gain unauthorized access to shared data log.
Mitigation	Ensure that standard authorization techniques are used to read and write data to the log.

Table 20: Region-2- Spoofing-threat

Type	Information Disclosure
Likelihood	Very likely
Impact	disclosure the data to illegal party
Priority	High
Description	An adversary can gain access to sensitive data by sniffing traffic to pipeline
Mitigation	Secure communication to the services using SSL/TLS.

Table 21: Region-2- Information Disclosure-threat

7 Prototype(Proof of Concept)

This chapter discusses the assumptions that would simplify some of solutions in order to demonstrate the concept of the architecture. It also emphasizes on the requirements for OMS to make the system as a whole. finally, it end ups with a review of the deployment plan.

7.1 Mocking OMS

Due to the reasons that have been mentioned in the architecture chapter, The OMS is mocked. The following discusses, the parts of it that have direct connexion to the context of this project, which are fetching, persisting, and what is the best approach to make this data available to the analysis service, these points will be taken into account when designing and implementing the service by BCT.
"When we create a software application, a large part of the application is not directly related to the domain, but it is a part of the infrastructure or serves the software itself[..]. However, when domain-related code is mixed with the other layers,it becomes extremely difficult to see and think about. Superficial changes to the UI can actually change business logic.To change a business rule may require meticulous tracing of UI code, database code, or other program elements. Implementing coherent, model-driven objects becomes impractical. Automated testing is awkward. With all the technologies and logic involved in each activity, a program must be kept very simple or it becomes impossible to understand. Therefore, partition a complex program into LAYERS. Develop a design within each LAYER that is cohesive and that depends only on the layers below".(Evan 2003*a*) Based on this, it is a good practice to design the object management system service that follows this approach.
As shown in the diagram, OMS compromise with many components. every component handles the complexity of one matter. The components that are intuitive are excluded from this discussion. However, this section focuses mainly on the component that may be unclear in addition to, the domain model and data model

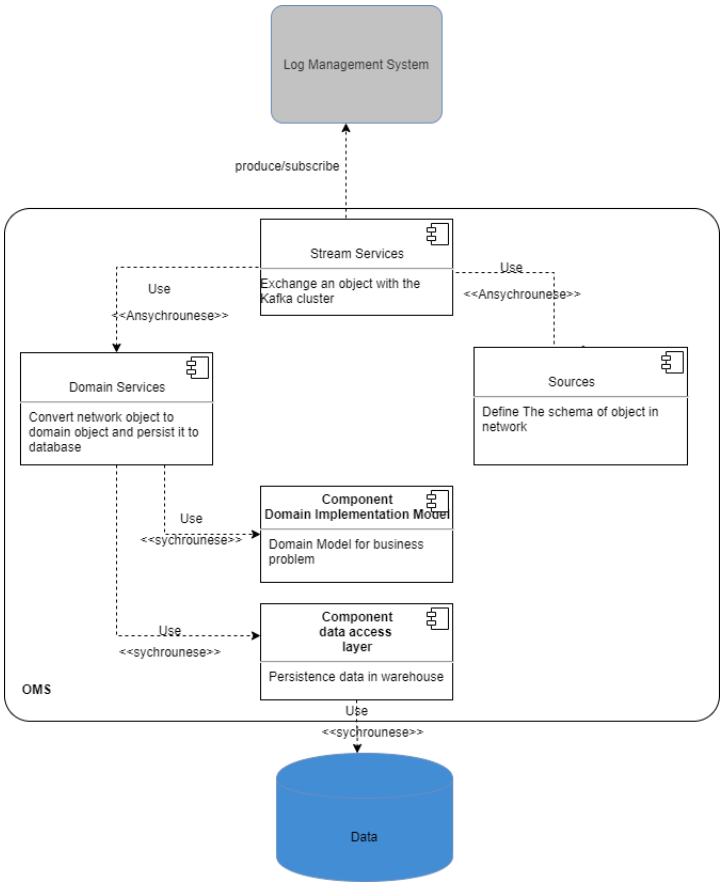

Figure 16: OMS Component Diagram

1. Sources
   "When you're working with a remote interface [..], each call to it is expensive. As a result you need to reduce the number of calls, and that means that you need to transfer more data with each call". (Fowler 2002).Consequently the schema of call could does not map any object in the domain model.

2. Domain Services
   Mainly used to fill the gap between source object and domain model object then , it uses the data access layer to persist the domain model object.

3. Stream services
   This component is consider as a requirement. in a high level overview it ,receives the message from IoT device, process it then , writes the outcome

to the new topic in the shared data log.

4. Domain Model
   As was pointed out earlier, the domain model problem is a building management matter. The building can be seen as a fixed asset, which, in turn, includes many other fixed assets" floors". The floor includes many other fixed assets "room". The room includes many properties(temperature, humidity). The temperatures and humidity are classified as a property as their values change according to the time. Accordingly, this is a hierarchy model. Each room is tracking its properties, similar to the floors and the buildings. The primary aim of this model is to ensure that every building complies with governance roles. As it states in the UML class diagram, the assets follow the composition pattern where floor and building are compositions; meanwhile, the room is the leaf. The composition pattern allows the client to iterate over all assets in the same way whether this asset is a leaf or root.(Evan 2003*b*)

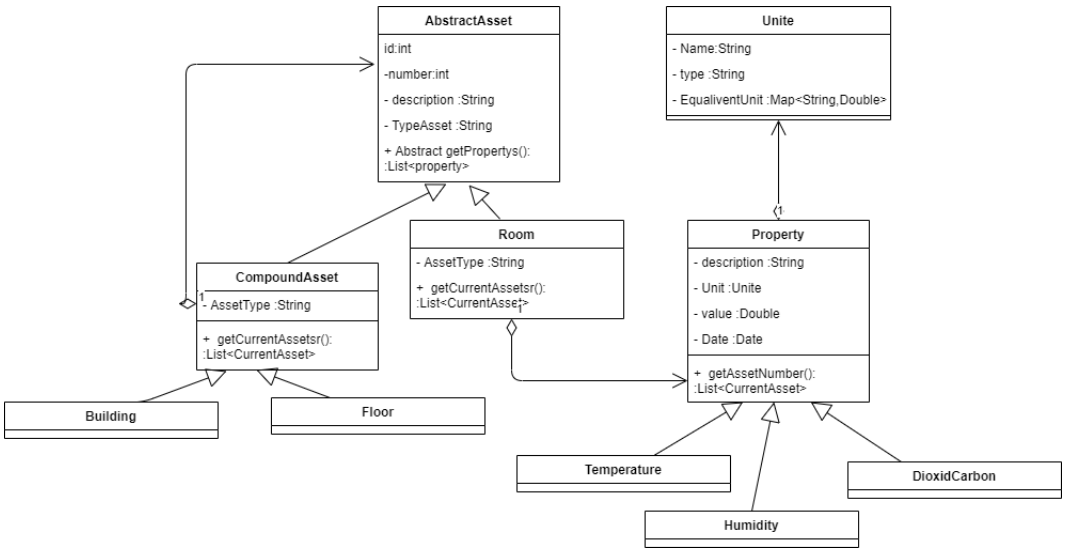

Figure 17: Domain model class diagram

5. Data Model
   The following diagram is an ER diagram that maps to the domain model. Designing the data model in this way gives the advantage of using polymorphism in the domain model; meanwhile, it does not repeat the same columns name. The diagram illustrates that the data model applies the join table strategy for inheritance meaning, each class of the inheritance hierarchy maps to its own database table. The table that maps to abstract superclass contains columns for all shared entity attributes whereas, the other tables hold only the columns specified for the mapped entity class and a primary key with the same value as the record in the table of the superclass.

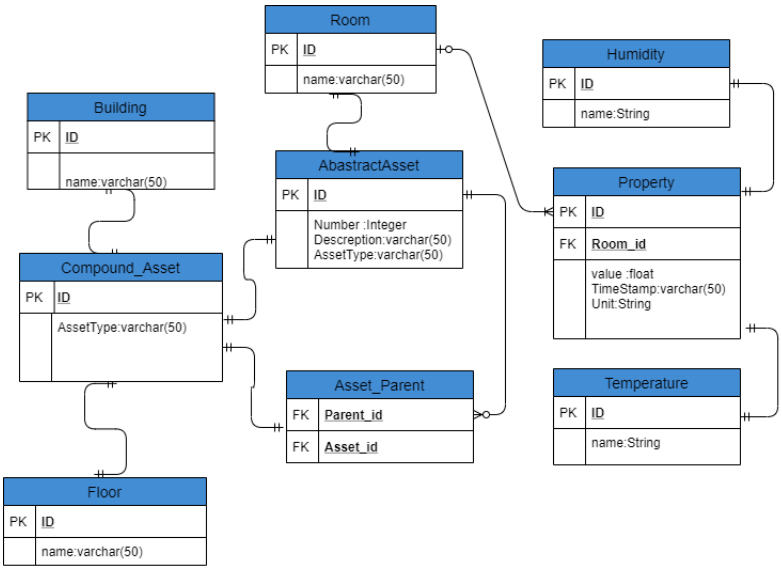

Figure 18: ER diagram

7.2 IoT device

In the production the IoT device should only send the Environment data(reduce the data in network). identifying the geographical data and the time should be a task to the OMS. But for the sake of simplicity, Each IoT device is aware of its geographical location within the building. it combines the location data besides the date when it sends the temperature and humidity to the data shared log.

7.3 Cloud gateway

The importance of the cloud gateway has been discussed, But as a assumption to the prove of the concept, the IoT data will be forwarded to the data shared log and skip the IoT cloud gateway.

7.4 Deployment view

This subsection explains how the prototype will be deployed to prove the concept. Both OMS and Data transformation will be containerized inside a docker container and Azure kubrenates will be the execution environment whereas, The IoT device will be out of the Azure kubernates cluster. the data log management system and data mart will be delivered by azure cloud service.

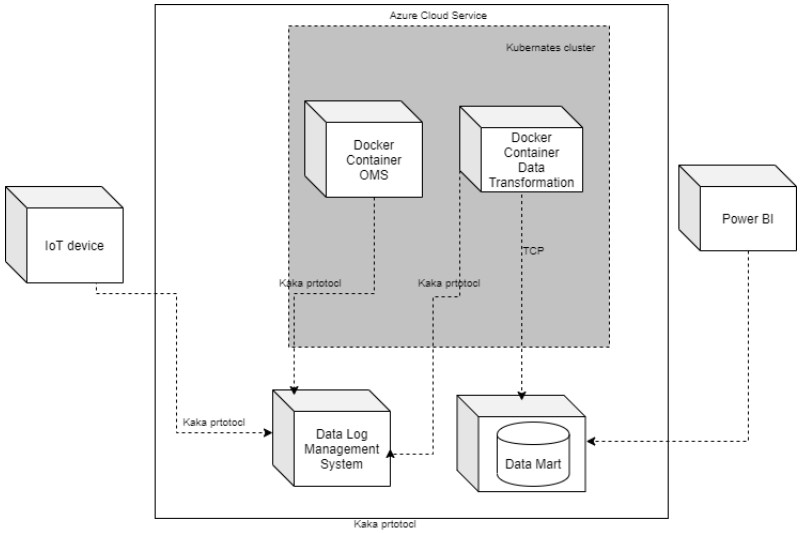

Figure 19: Deployment diagram

8 Implementation

This chapter discuses the implementation of the system. which technologies have been choose and the reasons behind the choices.

8.1 IoT device

Python high-level scripting language. It has useful libraries that support a wide range of sensors type. Getting data from sensors can be done with fewer lines of code thanks to these libraries. Moreover, cloud services like Google and Azure have SDK that supports this language. These SDK make sending the telemetry data to the cloud very straightforward with fewer lines of code. The following code shows how it is easy to get data from the sensor.

```
1 def get_Facts():
2     temperature,humidity = Adafruit_DHT.read_retry(DHT_SENSOR,
      DHT_PIN)
3         return temperature,humidity
```

Listing 1: Get Data From sensor

8.2 Cloud Gateway

In the architecture chapter, it was mentioned that the cloud gateway is Azure IoT hub, and we've seen that letting the cloud gateway writes the telemetry data to the Kafka cluster is the best approach to bring data to the solution. Azure IoT hub supports routing data to the Azure IoT event service. The way that the azure IoT event work is similar to the Kafka .it has the same notions. Moreover, it allows the consumer of the Kafka cluster to consume data in the same way without needing to change any lines of code.

8.3 OMS

This service uses the java spring framework thanks to the many reasons. We've seen what is the model-driven design mean and what is the values that can be achieved by following this approach. Spring framework helps to develop this approach by dividing each complexity into layers.

- Web layer: Concerns on the complexity of accessing the network. Defining the route and the sources that will be used over the network.
- Data access layer: Spring has a powerful way to handle the complexity of accessing the database by using the notion of JPA specification. The JPA specification allows defining which objects should be persisted, and how those objects should be persisted in the applications. By itself, JPA is not a tool or framework; instead, it defines a set of concepts that can be implemented by any ORM tool or framework like Hibernate. This increase the level of flexibility since it is possible to change the ORM tool

without requiring any change in the code.Tyson (2019) Moreover, using this API allows the developer to focus on the domain model problem without much concern about the persistence and retrieving objects from the database. All the developer has to do is add some annotations and let the underneath framework take care of the complexity . for instance in the prototype chapter the composition pattern has been used as a solution to the domain model. We have seen that the buildings and floors are the compositions whereas the rooms are the leaves. To make this pattern works although the data is stored in a database is, add some samples annotations to the composition class like following

```
1     @OneToMany(cascade = CascadeType.ALL, orphanRemoval = true
      ,fetch = FetchType.EAGER)
2     @JoinTable(name = "Asset_property", joinColumns = {
      @JoinColumn(name = "parent_id")}, inverseJoinColumns = {
      @JoinColumn(name = "asset_id")})
3     private final List<AbstractAsset> assets = new ArrayList
      <>();
4
5
```

Listing 2: JPA example

- It is easy to embed Kafka consumer and producer API to the framework. in-addition, to change the configuration according the environments(production,development).
- Relatively easy to preform tests.

8.4 Data Transformation

This service also uses the spring framework as a technology for the same reasons. Since the primary task of this service is to process the stream and perform aggregation, let's review some code and see how this code achieves the requirements.

```
1 KStream<String, State> stream = kStreamBuilder.stream("Analysis",
      Consumed.with(AppSerdes.String(),AppSerdes.State()).
      withTimestampExtractor(new StateTimeExtractor())
2         );
```

Listing 3: Creating source processor node

The code above creates the source of the topology. Since the broker sends the record as binary data and the record consists of key and message, we should provide the topology in how to deserialize the message and the key. Meanwhile, in most cases, the outcome of the topology is written to the new topic in the Kafka cluster. Consequently, we need the serializer. Thus, the Kafka stream API combines serializer and deserialize in one class, and it calls serdes.

```
KTable<Windowed<Integer>, WindowStatistic> KT0 = stream.groupByKey
    (Grouped.with(AppSerdes.Integer(), AppSerdes.IoTMessage())).
    windowedBy(TimeWindows.of(Duration.ofMinutes(2)).grace(Duration
    .ZERO))
```

Listing 4: Creating groupby key

As it states in the code above, after creating the source, we add two nodes to the topology. First one group by key. In the design chapter, we have decided that the key of the record is the room number. The second one is the time window.

```
.aggregate(
            () -> new WindowStatistic(),
            (k, v, aggValue) ->
                {
                    WindowStatistic bs = new WindowStatistic();
                    bs.setFloorNumber(v.getFloorNumber());
                    bs.setRoomNumber(v.getRoomNumber());
                    bs.setDate(v.getTime());
                    bs.setTotalTemperature(v.getTemp()+aggValue.
                    getTotalTemperature());
                    bs.setTotalHumidity(v.getHum()+aggValue.
                    getTotalHumidity());
                    bs.setStateCount(aggValue.getStateCount()+1);
                    if (v.getHum() > aggValue.getMaxHumidity())
                      bs.setMaxHumidity(v.getHum());
                    else
                      bs.setMaxHumidity(aggValue.getMaxHumidity());
                    if (v.getTemp() < aggValue.getMinTem())
                        bs.setMinTem(v.getTemp());
                    else
                      bs.setMinTem(aggValue.getMinTem());
                    if (v.getTemp() > aggValue.getMaxTem())
                      bs.setMaxTem(v.getTemp());
                    else
                      bs.setMaxTem(aggValue.getMaxTem());
                    if (v.getHum() < aggValue.getMinHumidity())
                      bs.setMinHumidity(v.getHum());
                    else
                      bs.setMinHumidity(aggValue.getMinHumidity());
                    return bs;
                },
              Materialized.<Integer, WindowStatistic, WindowStore
              <Bytes,byte[]>>with(AppSerdes.Integer(),
              AppSerdes.StateStatistic())
          )
```

Listing 5: Creating Aggregation node

The above code adds the third node to topology, which is the aggregation node. The aggregate method takes three functions as parameters. The first one to initialize the value, and it is called by framework one time when a new record that belongs to the new time window or a new room enters the stream. The second function returns bs object. The bs object will be passed as aggValue at the next call of this function. The framework calls this function when a new

record enters the stream, which belongs to the same time window and room to the last record. The parameters of this function are (k, v , aggvalue). “k” is the key for a new record, that belongs to the same time window and room of the last record. “Value”: The value for a new record, which belongs to the same time window and room of the last record. “aggrecord” is the value of the last record, which belongs to the same time window and room of the current record. The last function is used to access the state store. the state store saves the previous value of the aggevalue

```
/* suppres the value to emmit after the time window is finish  */
.suppress(Suppressed.untilWindowCloses(unbounded()));
```

Listing 6: suppress the topology to emit the value

The above code suppresses the topology to emit the value only when the time window is closed.

```
/*creating the sink which, is saving the record to the data mart*/
    KT0.toStream().foreach(
                (key, value) -> {
                   ioTMessageService.saveFact(value);
                    }
                );
        return stream;
    }
```

Listing 7: Creating sink node

Finally, the suppressed value is passed to the downstream then it is persisted to the data mart.

8.5 Security

Another crucial part on the system is security issues as we discussed in the threat model it is essential to the whole system to exchange data securely meaning the data needs to be encrypted and only the subsystem inside the solution can decrypt data. We’ seen one solution was to implement the connection over TLS. Thus let’s see how to implement it. Each service in the solution has a public-private key pair and a certificate to identify them, however, an intruder can claim to be any service in the solution since the certificate unsigned by a trusted party. Therefore, to prevent the counterfeiting, the certificate need to be singed by a legitimate authority. The official Kafka documentation use the following terminology to clarify the role of this legitimate authority(CA).” CA works like a government that issues passports - the government validates the identity of the person applying for the passport and then provides a passport in a standard form that is difficult to forge. Other governments verify the form is valid to ensure the passport is authentic. Similarly, the CA signs the certificates, and the cryptography guarantees that a signed certificate is computationally difficult to forge”. Thus, signing the certificate by CA assurances to any broker in the Kafka cluster that they are connecting to the authenticated service in the solution and the data is encrypted. Every service in the solution has ,a keystore to store its

identity , and truststore to store all the certificates that the service should trust. Applying this approach has difficulties in terms of scalability meaning, adding a new broker to the cluster entails, adding it as a trusted machine in every service on the solution. Alternatively adding a certificate into one's truststore means trusting all certificates that are signed by CA. This approach is called the chain of trust. In other words, by using chain of trust, It is possible to sign all certificates in the solution with a single CA, and have all services share the same truststore that trusts the CA.Apache-Software-Foundation (2017)

9 Testing and Validation

This section provides an overview of what need to be tested and how to carry out the tests.

9.1 Test Strategy

9.1.1 Feature to be tested

ID	Micro service	Component	Applicable Roles	Description
FT1	Power BI	–	Facility Manger	Facility manger: the facility manger can see the historical data(temperature,Humidity) for any assert(Floor,Room) at any point time in the past.
FT2	Data Transformation	Domain Services	System	The system can persist the data to the data mart.
FT3	Data Transformation	Stream processing	System	The system can process the stream and compute the average,max,min value .
NFT1	IoT device	NetworkServices	IoT device	IoT device:Only authenticated device can write to the log
NFT2	OMS, Data-Transformation	Stream processing	System	System:scaling up these services does not influence on the data consistency
NFT3	OMS, Data-Transformation	Stream processing	System	System:the overview performance of the system when handling the ioT message

Table 22: Feature to be tested

9.1.2 Feature not to be tested

- User Interface.
- Regression test.
- OMS database.
- User security.
- Dashboard performance.
- User acceptance testing.

These features has been excluded from the test strategy because, It does not influence on the prove of the concept to the architecture. other features like user security and dashboard performance have been excluded since ,the Microsoft power BI is a tool archive these features.

9.2 Test type

Manuel testing for dash board, Integration Testing (Individual software modules are combined and tested as a group),System Testing: Conducted on a complete, integrated system to evaluate the system's compliance with its non-functional requirements(security-scalability-performance,availability)

9.3 Test Objective

The test objectives are to verify the recommended architecture meets the functional and none functional requirements of the air quality system.

9.4 Test Criteria

This subsection clarifies when the test should be considered successful and when it is failed.

9.4.1 Failure Criteria

The test is considered to have failed when, one of the output does not meet the expected value.

9.4.2 Passed Criteria

The test is considered successful when, all the output meet the expected value.

9.5 Tools

Name	Description
Test-Driver-Topology	"simulates the library run-time that continuously fetches records from input topics and processes them by traversing the topology. You can use the test driver to verify that your specified processor topology computes the correct result with the manually piped in data records. The test driver captures the results records and allows to query its embedded state stores".Kafka Documentation (2019) .
test containers	"Testcontainers is a Java library that supports JUnit tests, providing lightweight, throwaway instances of common databases, Selenium web browsers, or anything else that can run in a Docker container".(North 2015)
EmbdedKafka	mock the kafka broker in the memory.

Table 23: test tools

9.6 Test Environment

The Test Environment should be setup as figure below

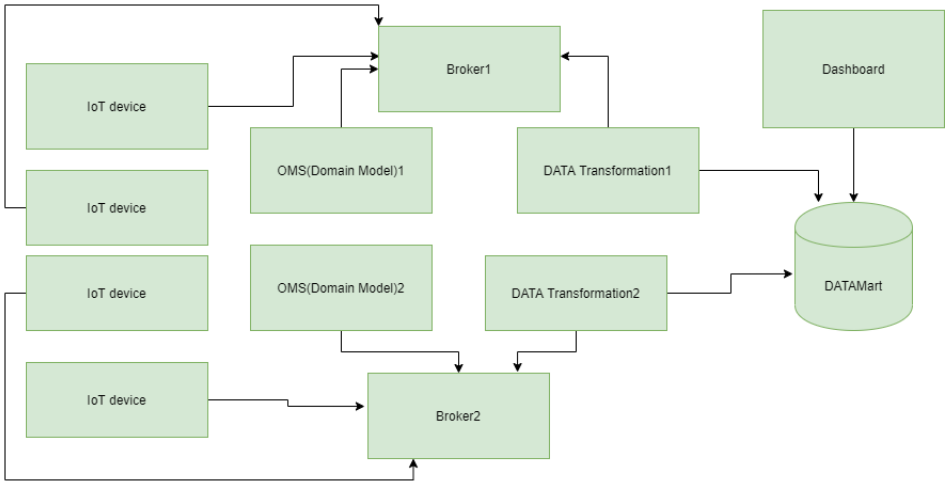

Figure 20: Test Environment

9.7 Test cases

This subsection explains how to perform tests. These test cases indicate to the features that need to be tested as it is mentioned at "inside the scope subsection". Each feature is tested in more than possible scenario. Each scenario has its inputs and expected outputs. The test cases start with how to test functional requirements; then, it moves to non-functional requirements. The order of testing functional requirements(FT1, FT2, FT3) is very important, as the outputs of the previous test cases are the input of the current one. The way that the table uses to indicate to its inputs is, matching its scenario number with the scenario number of the previous table then, corresponding expected outputs are the inputs of the table.

1. (floor1,room1,10°,40%, 2018-04-20T06:15:24).
2. (floor1,room1,15°,35%, 2018-04-20T10:23:22).
3. (floor1,room1,22°,22%, 2018-04-20T15:23:22).
4. (floor1,room1,23°,22%,2018-04-20T16:17:45).
5. (floor1,room2,23°,21%,2018-05-26T16:17:45).
6. (floor4,room3,23°,21%, 2019-04-21T16:17:45).

The main reason for choosing these inputs is , these inputs represent all possible situations. Same day but different part of the day . two different days, two different months, two different years. Furthermore three different rooms two of them belong to the same floor and last one belongs to different floor.

9.7.1 FT3

One important task in data transformation service is to compute the average,max and min value for every room during 6 hours time window and emit the value to the down stream pipeline

S.No	scenario	data input	expected output	status
1	two records belong to the same room and time window	(1,2)	fact(floor1,room1,12.5°,37.5%, 2018-04-20-2)	pass
2	Four records belong to the same room and different time window	(1,2,3,4)	1. fact(floor1,room1,12.5°,37.5%, 2018-04-20-2)) 2. fact(floor1,room1,22°, 22%, 2018-04-20-3)	pass
3	two records belong to the different room and different time window	(5,6)	1. fact(floor1,room2,23°, 21%, 2018-05-26-3) 2. fact(floor4,room3,235°, 21%, 2019-04-21-3)	pass

Table 24: Test Case FT3

9.7.2 FT2

S.No	Scenario	Expected Output	Status
1	Stream processing emits one result belongs to the same room. Then, the domain service persists it to the data mart. The expected value represents the objects after have been retrieved from the data mart	1. date1=(2018,4,20,2) 2. floor1=(1,"first floor at bct") 3. room1=(1,"It room") 4. factEnviroment1= ("tempreature") 5. factEnviroment2= ("Huimidity") 6. fact1=(date1,floor1, room1, factEnvirment1,12.5°,) 7. fact2=(date1,floor1, room1, factEnvirment2, 37%)	pass
2	Stream processing emits two results belongs to the same room but different time window then, the domain service persists it to the data mart. The expected value represents the objects after have been retrieved from the data mart.	1. date2=(2018,4,20,2),date3=(2018,4,20,3) 2. floor1=(1,"first floor at bct") 3. room1=(1,"It room") 4. factEnviroment1= ("tempreature") 5. factEnviroment2= ("Huimidity") 6. fact3=(date2,floor1, room1, factEnvirment1,12.5°) 7. fact4=(date2,floor1, room1, factEnvirment2,37%) 8. fact5=(date3,floor1, room1, factEnvirment1,22°) 9. fact6=(date3,floor1, room1, factEnvirment2,22%)	pass

Table 25: Test Case FT2 part1

S.No	Scenario	Expected Output	Status
3	Stream processing emits two results belong to the different room and time window. Then, the domain service persists it to the data mart. The expected value represents the objects after have been retrieved from data mart.	1. date4=(2018,05,26,3), date5=(2019,4,21,3) 2. floor1=(1,"first floor at bct"),floor4=(4,"fourth floor at bct") 3. room2=(2,"Facility manger room"), room3=(3,"manger room") 4. factEnviroment1= ("tempreature") 5. factEnviroment2= ("Huimidity") , 6. fact7=(date4,floor1,room2, factEnvirment1,23°) 7. fact8=(date4,floor1, room2, factEnvirment2, 21%) 8. fact9=(date5,floor4,room3, factEnvirment1, 23°) 9. fact10=(date5,floor4, room3, factEnvirment2, 21%)	pass

Table 26: Test Case FT2 part2

9.7.3 FT1

Passing the dashboard test means that the schema of data mart is correct. We have seen that the previous tests that focused on testing the data from the moment it arrived at the data transformation service to the moment when data is persisted to the data mart have passed. Passing all of these tests means the system fulfills its functional requirements. The following are the inputs for this test after rearranging them. As a reminder, these inputs are the outputs from the previous test.

1. fact3=(date2,floor1, room1, factEnvirment1 , 12.5°)
2. fact4=(date2,floor1, room1, factEnvirment2 , 37%)
3. fact5=(date3,floor1, room1, factEnvirment1 , 22°)
4. fact6=(date3,floor1, room1, factEnvirment2 , 22%)
5. fact7=(date4,floor1,room2, factEnvirment1 , 23°)

6. fact8=(date4,floor1, room2, factEnvirment2, 21%)
7. fact9=(date5,floor4,room3, factEnvirment1, 23°)
8. fact10=(date5,floor4, room3, factEnvirment2, 21%)

S.No	Scenario	Expected Output	Status
1	facility manager wants to see all facts in all rooms for the 2018 year, the average(temperature, humidity) and min, max value during that year.	1. the records is map to same number of the input (1,2,3,4,5,6) 2. temperature: avg=19.16, max=23 , min=12.5 3. Humidity: avg=26.6, max=37 , min=21	pass
2	The facility manager wants to see all facts in all rooms for the April of the 2018 year, the average(temperature, humidity) and min, max value during that month.	1. the records is map to same number of the input (1,2,3,4) 2. temperature: avg=17.25, max=22 , min=12.5 3. Humidity: avg=29.5, max=37 , min=22	pass
3	The facility manager wants to see all facts in all rooms for the day 20th April at the 2018 year, the average(temperature, humidity) and min, max value during that day.	1. the records is map to same number of the input(1,2,3,4) 2. temperature: avg=21, max=22 , min=20 3. Humidity: avg=22.5, max=23 , min=22	pass
4	The facility manager wants to see all facts in all rooms for a floor1 the average(temperature, humidity) and min, the max value for that floor.	1. the records is map to same number of the input (1,2,3,4,5,6,7,8) 2. temperature: avg=19.16, max=23 , min=12.5 3. Humidity: avg=26.6, max=37 , min=21	pass

Table 27: Test Case FT1

9.7.4 NFT1

In the design chapter, an important point is indicated, which is the correction of data during the scalability. this test verifies that room's messages are always consumed by the same instance of data transformation

S.No	scenario	data input	expected output	statu
2	Four records each two of them belong to the different room	1. (floor1,room1,20°,23%, 2018-04-20T12:21). 2. (floor1,room1,22°,22%, 2018-04-20T16:04). 3. (floor1,room2,12°,16%, 2018-04-20T20:21). 4. (floor1,room2,10°,14%, 2018-04-20T23:04)	1. if the instance 1 consumes 1,2 , the instance 2 consume 3,4 2. if the instance 1 consumes 3,4 , the instance 2 consume 1,2	pass

Table 28: Test Case NFT1

9.7.5 NFT2

:

S.No	scenario	data input	expected output	statu
2	record come form authenticated IoT device	1. (floor1,room1,20°,23%, 2018-04-20T12:21).	1. The record is written to the log in the broker	pass
2	record come from unauthenticated device	1. (floor1,room1,20°,23%, 2018-04-20T12:21).	1. The record is not written to the log in the broker	pass

Table 29: Test Case NFT2

9.7.6 NFT3

This test describes how the overall performance of the application is validated and what are the metric need to be measure. lets' start in what are metrics that need to be measured

- The average time to poll record from log. It called AvgPull in the Performance metrics table .
- The average number of records processed per millisecond.
- The memory consumption.
- The cpu consumption.

Monitoring these metrics gives a good overview to diagnose the neck bottle problem and then make the right decision to improve the performance. Fortunately, Kafka Streams provides the mechanism for collecting these performance metrics. Then it reports them to Java Management Extensions (JMX). One thing to keep in mind is that JMX only works with live running applications, so the metrics we'll look at will be when the application is running. The metrics of

DTNum	Cpu	Memory	AvgPull	AvgRcs
1	1.6	150MB	2Milisece	1.75 records per Milliseconds

Table 30: Performance metrics with 1 IoT device

DTNum	Cpu	Memory	AvgPull	AvgRcs
1	3	250MB	4Milisece	3records per Milliseconds

Table 31: Performance metrics with 3 IoT

DTNum	Cpu	Memory	AvgPull	AvgRcs
1	1.6	150MB	2Milisece	1.75 records per Milliseconds
2	1.8	156	2.6Milisece	2.3 records per Milliseconds

Table 32: Performance metrics With 4IoT device

the table illustrates that the load is distributed equally among all microservice instance in every time a new IoT device is added.

10 Conclusion

Referring back to the main research question, which is about the recommended architecture for a system that collects IoT data to perform operational tasks. Then store all of that data for later analysis. This analysis helps the decision-maker to find out trends and consequently make strategic decisions that depends on facts. The first step was to choose the micro-services architecture. Decoupling, flexibility, deployability are essential factors in making this decision. Another critical decision is shared data log, which, besides the micro-services architecture, defines the general architecture of the system. The presence of IoT devices that consistently produce immutable data was the main reason for choosing the data shred log. Then the decision was made to use Kafka protocol as a communication mechanism among micro-services. Scalability and fault tolerance are the reasons for choosing the Kafka protocol. In conclusion, The combination of micro-services, data shred log, and Kafka protocol makes the digital system mapping the dynamic nature of the system in the real world, mainly since the system deals with sensors that observe the changes in the environment around us. Sometimes these changes need a real-time response as they occur. In contrast, the absence of any element of this combination causes a gap between reality and the digital world. This gap because of the static nature of the digital system. This architecture is extendable not modifiable in other words, adding a new micro-service to achieve a new requirements does not effect on any other micro-service and there is no need to modify the architecture.Another advantages of this architecture is the natural technology meaning the new micsro-service could be implemented buy using any programming language.

References

Apache-Software-Foundation (2017), 'Kafka 2.5 documentation'.
URL: *https://kafka.apache.org/documentation/security*

Balkenende, M. (2018), 'The big data debate: Batch versus stream processing'.
URL: *https://thenewstack.io/the-big-data-debate-batch-processing-vs-streaming-processing*

Brown, S. (2019), 'Visualising software architecture with the c4 model'.
URL: *https://www.youtube.com/watch?time*$_c$*ontinue* $= 6v = x2 - rSnhpw0gfeature = emb_logo$

Burns, B. (2018), Designing distributed systems, O'Reilly Media, Inc, pp. 22–23.

Chaudhuri, S. & Dayal, U. (1997), 'An overview of data warehousing and olap technology', *ACM SIGMOD Record* pp. 5–8.

cofluent (2019), 'Valuable resources on stream processing'.
URL: *https://docs.confluent.io/current/streams/architecture.html*

Dunko, G., Misra, J., Robertson, J. & Snyder, T. (2017), 'A reference guide to the internet of things', *Bridgera* **2017**, 34–57.

Evan, E. (2003*a*), *Domain-Driven Design: Tackling Complexity in the Heart of Software*, Addison-WesleyProfessional.
URL: *http://books.google.com/books?id=W-xMPgAACAAJ*

Evan, E. (2003*b*), *Head first*, Addison-WesleyProfessional.
URL: *http://books.google.com/books?id=W-xMPgAACAAJ*

Fowler, M. (2002), Patterns of enterprise application architecture, Addison-Wesley Professional, 2003, p. 401.

Goswami, S. (2018), 'Scalability of kafka messaging using consumer groups'.
URL: *https://blog.cloudera.com/scalability-of-kafka-messaging-using-consumer-groups*

Hernandez, A. (2018), 'Bi data modeling with sqldbm'.
URL: *https://blog.sqldbm.com/bi-data-modeling-with-sqldbm/*

Kafka Documentation, D. (2019), 'Official kafka api documentation'.
URL: *https://kafka.apache.org/20/documentation/streams/developer-guide/testing.html*

Kleppmann, M. (2016), Making sense of stream processing, Confluent, pp. 41–52.

Kleppmann, N., Sshapira, G. & Palino, T. (2017), Kafka: The definitive guide real-time data and stream processing at scale, Confluent, pp. 29–33.

microsoft (2018), Microsoft azure iot reference architecture, Microsoft, pp. 13–15.

Myers, P., Saxton, A., Blythe, M. & Sharkey, K. (2019), 'Understand star schema and the importance for power bi'.
URL: *https://docs.microsoft.com/en-us/power-bi/guidance/star-schema*

Narkhede, N., Sshapira, G. & Palino, T. (2017), Kafka: The definitive guide real-time data and stream processing at scale, Confluent, pp. 13–15.

Network, P. (2018), 'Three dimensions of distributed system scalability design'.
URL: *https://medium.com/@Pointnity$_N$etwork/three − dimensions − of − distributed − system − scalability − design* − 8*e*0319163*c*8*d*

North, R. (2015), 'Test container java'.
URL: *https://www.testcontainers.org*

RADACAT-Team (2016), 'Data preparation; first and foremost important task in power bi'.
URL: *https://radacad.com/data-preparation-first-and-foremost-important-task-in-power-bi*

Richardson, C. (2019), Microservices patterns, Manning Publications Co, pp. 2–5.

Sethi, P. & Smruti, R. (2017), 'Internet of things: Architectures, protocols, and applications', *Journal of Electrical and Computer Engineering* **2017**, 11–17.

Shahan, R., Meadows, P. & Lamos, B. (2018), 'Internet of things (iot) security architecture'.
URL: *https://docs.microsoft.com/en-us/azure/iot-fundamentals/iot-security-architecture*

Storm, C. (2011), *Specification and Analytical Evaluation of Heterogeneous Dynamic Quorum-Based Data Replication Schemes.*

Tyson, M. (2019), 'Introduction to the java persistence api'.
URL: *https://www.javaworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html*