Instituto Superior de Engenharia do Porto

Smart IoT Service Builder Platform

Filipe Cruz

A dissertation submitted in partial fulfillment of
the requirements for the degree of Master of Science,
Specialisation Area of Software Engineering

Supervisor: Dr. Nuno Silva

Porto, July 5, 2022

Dedicatory

TODO

Abstract

Today there are more smart devices than people. According to **statista-number-devices** the number of devices worldwide is forecast to almost triple from 8.74 billion in 2020 to more than 25.4 billion devices in 2030.

The Internet of Things (IoT) is the connection of millions of smart devices and sensors connected to the Internet. These connected devices and sensors collect and share data for use and evaluation by many organizations. Some examples of intelligent connected sensors are: GPS asset tracking, parking spots, refrigerator thermostats, soil condition and many others. The limit of different objects that could become intelligent sensors is limited only by our imagination. But this devices are mostly useless without a platform to analyse, store and present the aggregated data.

Recently, several platforms have emerged to address this need and help companies/governments to increase efficiency, cut on operational costs and improve safety. Sadly, most of this platforms are tailor made for the devices that the company offers. This dissertation presents a platform and its development that assembles multiple services related to IoT into a single application. All the services provided by this platform attempt to be sensor-neutral and are to be exhibited under the same unified application.

Keywords: Internet of Things, Stream Processing, Big Data, Configurability, Real Time Systems

Resumo

Atualmente, existem mais sensores inteligentes do que pessoas. De acordo com **statista-number-devices**, o número de sensores em todo o mundo deve quase triplicar de 8,74 bilhões em 2020 para mais de 25,4 bilhões em 2030.

O conceito de IoT está relacionado com a interacção entre milhões de dispositivos inteligentes através da Internet. Estes dispositivos e sensores conectados recolhem e disponibilizam dados para uso e avaliação por parte de muitas organizações. Alguns exemplos de sensores inteligentes e seus usos são: dispositivos GPS para rastreamento de activos, monitorização de vagas de estacionamento, termostatos em arcas frigoríficas, condição do solo e muitos outros. O número de diferentes objectos que podem vir-se a tornar sensores inteligentes é limitado apenas pela nossa imaginação. Mas estes dispositivos são praticamente inúteis sem uma plataforma para analisar, armazenar e apresentar os dados por eles agregados.

Recentemente, várias plataformas surgiram para responder a essa necessidade e ajudar empresas/governos a aumentar a sua eficiência, reduzir custos operacionais e melhorar a segurança dos espaços e negócios. Infelizmente, a maioria dessas plataformas é feita à medida para os dispositivos que a empresa em questão oferece. Esta tese apresenta uma plataforma que permite a criação e agregação de vários serviços relacionados com IoT num ambiente único. Todos os serviços fornecidos por esta plataforma procuram ser agnósticos em relação aos dispositivos inteligentes suportados.

Acknowledgement

TODO

Contents

List of Figures

List of Tables

List of Algorithms

List of Source Code

List of Symbols

a	distance	m
P	power	W ($\mathrm{J\,s^{-1}}$)
ω	angular frequency	rad

Chapter 1

Introduction

1.1 Problem

1.2 Context

1.3 Approach

1.4 Objectives

1.5 Achieved Results

1.6 Document Structure

Chapter 2

State of the Art

2.1 Internet of Things

2.1.1 Brief Description

2.1.2 Practical Applications

2.1.3 Enterprise Challenges

2.1.4 Renowned Solutions

2.2 Big Data

2.2.1 Brief Description

2.2.2 Challenges

2.3 Synopsis

Chapter 3

Analysis

3.1 Business Analysis

3.1.1 Fleet Management

3.1.2 Smart Irrigation

3.1.3 Fire Outbreak Surveillance

3.2 Technical Analysis

3.2.1 Data Aggregation

3.2.2 Data Filtering

3.2.3 Data Storage

3.2.4 Data Transformation

3.2.5 Data Analysis

3.2.6 Data Presentation

3.2.7 Trigger Warning System

3.2.8 User Authentication/Authorization

3.3 Synopsis

Chapter 4

Requirements Elicitation

4.1 Functional Requirements

4.2 Non Functional Requirements

4.3 Synopsis

Chapter 5

Design

This section goal is to describe the overall system design to the reader. First the reference architectures used for this project will be presented. Then the various system scopes will be introduced, followed by section regarding the domain model. After this the system's architectural design will be presented and major decisions/alternatives discussed. At last, a synopsis of this chapter can be read.

5.1 Reference Architecture

5.2 System Scopes

The system designed can be divided is three main scopes as disclosed in the Figure 5.1.

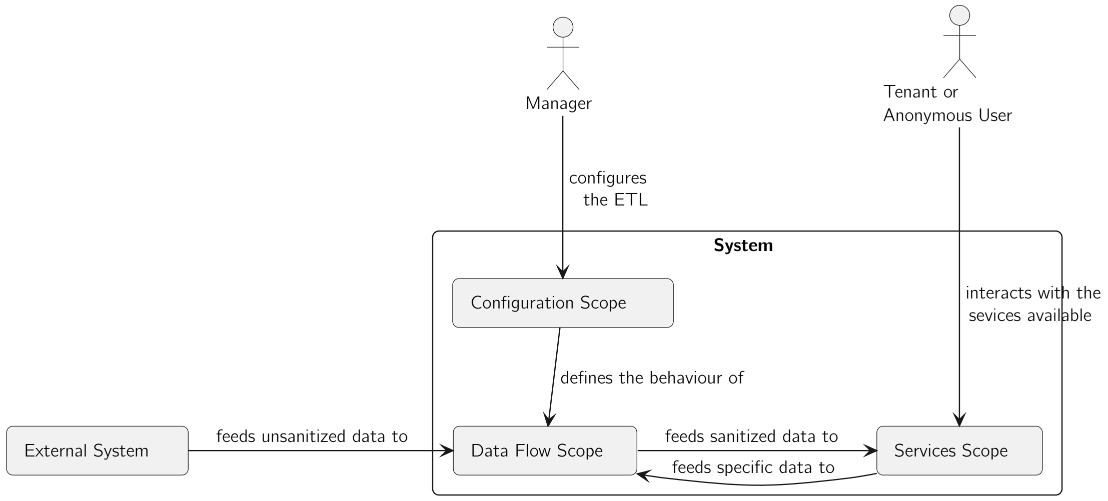

Figure 5.1: System Scopes

The **Configuration Scope** adheres to the configuration and visualization of internal processes/contexts. This processes, such as: (i) data decoders, (ii) device inventory, (iii) warning rules definition and (iv) device ownership, are related to the **Data Flow Scope**. It is also possible to manage tenants' access and permissions in the system in this scope.

The **Data Flow Scope** behaves according to what is defined in the **Configuration Scope** and acts as a pipeline where raw device data goes though various stages till it is sanitized and ready to be supplied to the **Services Scope**. The **Data Flow Scope** is where internal processes occur, such as: (i) data transformation, (ii) data enrichment, (iii) data validation, (iv) data ownership clarification and (v) warnings dispatching.

The **Services Scope** is comprised of services that present and act according to the sanitized data that was supplied to them. This services applicability range from (i) smart irrigation, (ii) fleet management, (iii) fire detention, (iv) physical security access monitoring, (v) air quality monitoring and anything else deemed interesting.

5.2.1 Configuration Scope

The **Configuration Scope** is responsible for managing the following contexts:

- **Data Processor**: manages simple data mappers;
- **Data Decoder**: manages scripts to transform data;
- **Device Management**: manages device information such as name, metadata, static data and other notions;
- **Identity Management**: manages device ownership and users permissions;
- **Rule Management**: manages scripts that consume device data and produce alerts.

Each context allows an authorized user to manage its resources, e.g. the data processor context manages the creation, deletion and renovation of data mappers.

This operations require various verifications, alter the system internal state and are therefore prolonged operations.

5.2.2 Data Flow Scope

The **Data Flow Scope** is responsible for processing incoming data according to what is defined in the **Configuration Scope**. Both scopes share the same contexts, apart from the data validation context (only present in this scope).

The data validation context preforms basic data filtering based on static rules, e.g. battery percentage reported has to be in between 0 and 100.

This scope applies changes to the device data that flows though the system. This changes are stateless and don't change the overall state of the internal system state.

This scope was decoupled from the **Configuration Scope** even though they both work with the same contexts. The decision was taken based on the pretext that despite the similarities in context the operation/business processes of this two scopes were conflicting.

The **Configuration Scope** requires scarce but heavy computations that alter the internal system state while the **Data Flow Scope** requires plentiful but light computations that don't alter the internal system state as summarized in the Table 5.1.

Due to this discrepancy it's expected for each scope to have different requirements regarding horizontal scaling. With the addition of more devices to the platform, and subsequently higher ingress volume, **Data Flow Scope** will need to scale. Since the **Configuration Scope** is intended mostly for the manager of the platform, a small user pool, the need to scale is smaller.

Comparison of Operations	Configuration Scope	Data Flow Scope
Alter internal system state	yes	no
Alter sensor data	no	yes
Required computation power/time	high	low
Frequency of usage	low	high

Table 5.1: Comparison of Operations in Data Flow and Configuration Scopes

5.2.3 Service Scope

The **Service Scope** is responsible for presenting Internet of Things (IoT) business cases to end users. This scope is comprised of services that consume and publish data to **Data Flow Scope**. Currently, as an Minimal Value Product (MVP) the following business cases are:

- **Fleet Management**: basic service to monitor a fleet of cars regarding their location;
- **Smart Irrigation**: service to automate and monitor the irrigation of zones based on sensor readings;
- **Notification Management**: service to view and manage the delivery of triggered alerts.

Each service is bounded to what type of data receives and sends back to the **Data Flow Scope** as detailed in Sections 5.3.1 and 5.3.2.

5.3 Domain

5.3.1 Concepts

5.3.2 Shared Model

5.3.3 Bounded Contexts

5.4 Architectural Design

In order to describe the system in detail at the architectural level, an approach based on the combination of two models, C4 (Brown 2018b) and 4+1 will be followed.

The 4+1 View Model (By and Jiang 1995), proposes the description of the system through complementary views thus allowing to separately analyze the requirements of various software stakeholders, such as users, system administrators, project managers, architects, and programmers.

The five views are thus defined as follows:

- **Logical view**: relative to the aspects of the software aimed at responding to business challenges;
- **Process view**: relative to the process flow or interactions within the system;
- **Development view**: relative to the organization of the software in its development environment;
- **Physical view**: relative to the mapping of the various components of the software in hardware, i.e. where the software is executed;

- **Scenario view**: related to the association of business processes with actors capable of triggering them.

The C4 Model (Brown 2018b, Brown 2018a) advocates describing software through four levels of abstraction: (i) system, (ii) container, (iii) component, (iv) code. Each level adopts a finer granularity than the level that precedes it, thus giving access to more details of a smaller portion of the system. These levels can be likened to maps, e.g. the system view corresponds to the globe, the container corresponds to the map of each continent, the component view corresponds to the map of each of each country, and the code view to the map of roads and neighborhoods in each city.

Different levels allow you to tell different stories to different audiences.

The levels are defined as follows:

- **Level 1**: Description (context) of the system as a whole;
- **Level 2**: Description of system containers;
- **Level 3**: Description of components of the containers;
- **Level 4**: Description of the code or smaller parts of the components.

These two models can be said to expand along distinct axes, with the C4 Model presenting the system with different levels of detail and the 4+1 View Model presents the system from different perspectives. By combining the two models it becomes possible to represent the system from several perspectives, each with various levels of detail. To visually model/represent the ideas designed and alternatives considered, the Unified Modeling Language (UML) was used.

In the following sections only combinations of perspectives and level deemed relevant for the design of the solution are presented.

The C4 level 4, code, will not be exhibited.

5.4.1 C4 Level 1 - Context

The context level aims at introducing the system as a whole. The external systems and users that communicate/interact with the system, **Sensae Console**, are demonstrated. Throughout this section the relevant C4 views of level 1 (context level) are presented.

Context Level - Logical View

The logical view of the system is introduced here, complete but not detailed, in order to answer the use cases and requirements discussed in *****TODO*****. This takes into account the interactions of the platform with external systems and its interaction with the various actors of the system (Figure 5.2).

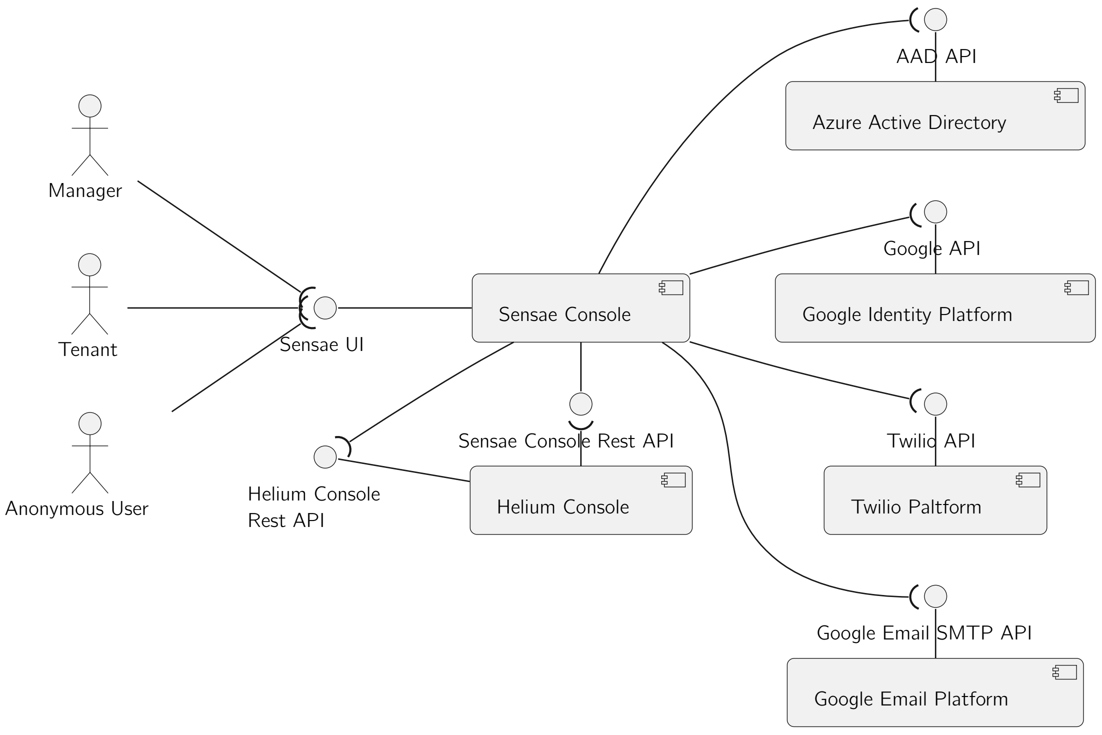

Figure 5.2: Context Level - Logical View Diagram

The external systems and its functions are as follows:

- **Helium Console**: Device data hub;
- **Azure Active Directory**: User authentication/identity;
- **Google Identity Platform**: User authentication/identity;
- **Twilio Platform**: SMS delivery;
- **Google Email Platform**: Email delivery.

The reason behind the use of external authentication/identity services is described in the Section 5.5.2.

Context Level - Physical View

Next is the physical view (Figure 5.3), intended to familiarize the reader with the idealized production environment.

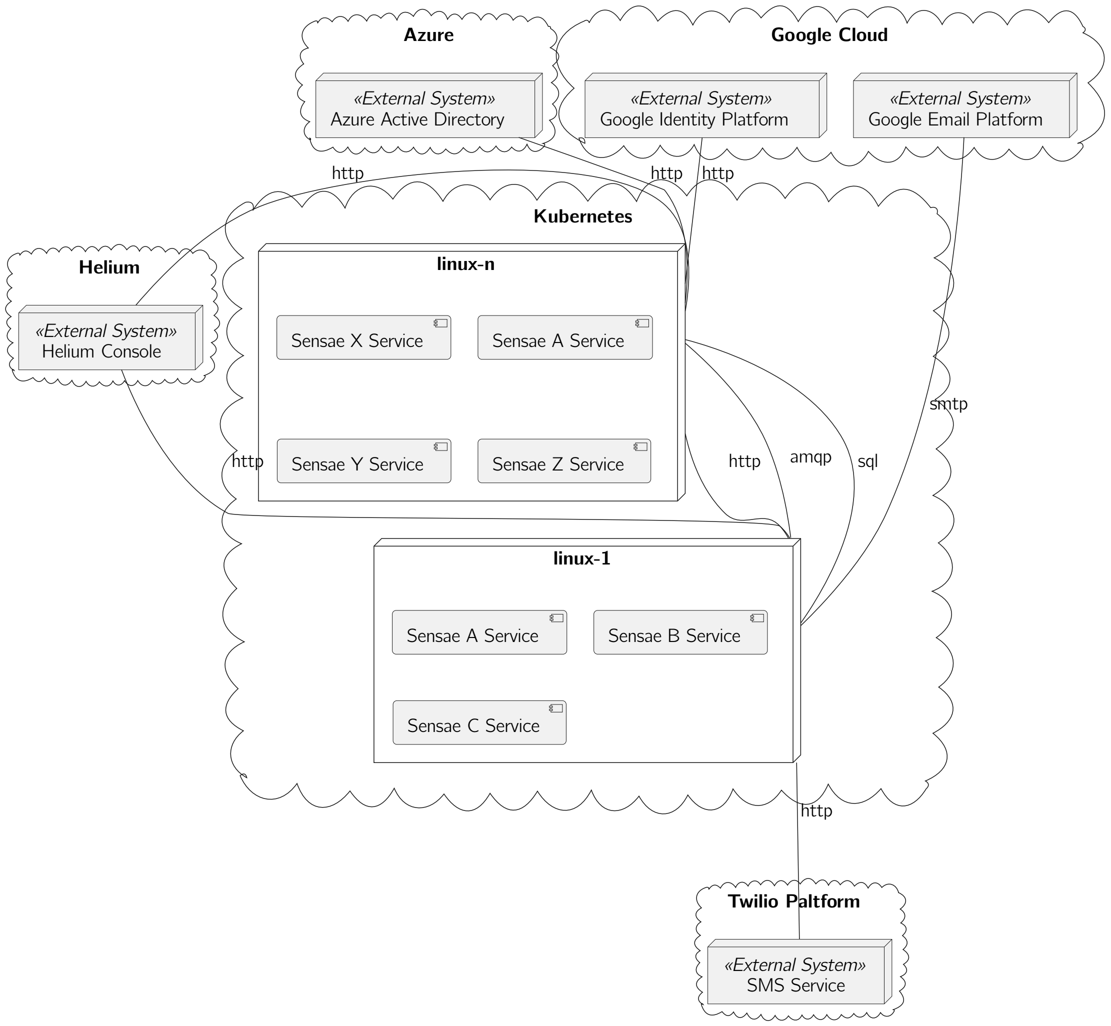

Figure 5.3: Context Level - Physical View Diagram

Due to time constrains the environment was not deployed to *Kubernetes* and the solution is instead orchestrated using *Docker Compose* in a single node/server.

Context Level - Development View

Next is the development view (Figure 5.4), intended to familiarize the reader with how the software is organized.

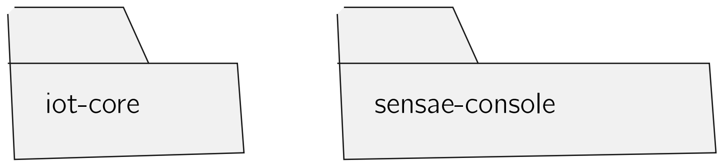

Figure 5.4: Context Level - Development View Diagram

The package *iot-core* contains the shared model discussed in the Section 5.3.2, and functions to define what type of device data/internal state a backend containers wants to subscribe or publish (discussed in Section 5.3.1).

The package *sensae-console* contains software of the various containers needed to run the **Sensae Console**. As expected *iot-core* is a core dependency for the *sensae-console* backend containers.

Context Level - Synopsis

The process view was not represented since at this level the interactions between the system, actors and external systems, are too abstract to be relevant for the reader.

5.4.2 C4 Level 2 - Containers

5.4.3 C4 Level 3 - Components

5.5 Architectural Alternatives Discussed

5.5.1 Data Streaming/Pipeline

5.5.2 User Authorization/Authentication

5.5.3 Internal Communication

5.6 Synopsis

Chapter 6

Implementation

6.1 Technical Decisions

6.2 Technical Description

6.3 Testing

6.4 Continuous Integration/Continuous Delivery

6.5 Synopsis

Chapter 7

Evaluation of the Solution

7.1 Approach

7.2 Subjective Critique Evaluation - Configuration View

7.3 Subjective Critique Evaluation - Operation View

7.4 Synopsis

Chapter 8

Conclusion

8.1 Achievements

8.2 Unfulfilled Results

8.3 Future Work

8.4 Synopsis

Bibliography

Brown, Simon (June 2018a). *The C4 Model for Software Architecture*. [Online; accessed 30. Jun. 2022]. url: `https://www.infoq.com/articles/C4-architecture-model/`.

– (2018b). *The C4 model for visualising software architecture*. [Online; accessed 30. Jun. 2022]. url: `https://c4model.com`.

By, Slides and Jack ZhenMing Jiang (Nov. 1995). "Architectural Blueprints–The "4+ 1" View Model of Software Architecture". In: [Online; accessed 30. Jun. 2022].

Appendix A

Appendix Title Here

Write your Appendix content here.