**IoT Use Case: Smart Container**

# Overview

The smart container serves as a critical component within the cold chain logistics system, specifically tailored for temperature-sensitive transportation scenarios. Its primary goal is to deliver a steady stream of accurate, real-time data with minimal delays to external applications. This data encompasses live readings of temperature, humidity, and precise GPS coordinates.

For instance, consider a situation involving blood transportation: At Groote Schuur Hospital in Cape Town, there arises an urgent need for a patient requiring a blood type O transfusion. Unfortunately, the hospital's current supply falls short. In response, they promptly reach out to their closest blood bank for an emergency provision. The blood bank, recognizing the necessity for a controlled environment, entrusts the blood to our smart container, ensuring it is maintained at a precise temperature of 6°C. A dedicated driver is dispatched for the timely delivery.

* Blood transport under right conditions 🡪 monitoring
* Find the shortest distance and suitable container (next: which ice is enough for the transport) 🡪 rout optimization
* Safe over hand of goods

Throughout the transportation process, both the hospital and the blood bank are empowered to closely monitor the container's whereabouts and the current temperature status. This level of real-time tracking and data accuracy ensures the safety and efficacy of this vital medical delivery.

# Technical Requirements and Boundary conditions

## Requirements / User Stories

* Route Optimization
  + Find the suitable Driver (algo)
  + Find the shortest way
* Visualization
  + of real-time position track for client and admin
  + current temperature in the cooler box
* Notifications
  + e.g via sms or mobile app when temperature drastically changes
  + e.g when destination is reached
* Safe over-hand
  + pick up goods from driver = right driver and right good
  + deliver to customer = right good to right customer)

Security Aspects:

* Transportation layer: private cluster for access via Broker, no public access to subscribe for data
* DB access only from server (user, pwd on server side)
* Access to server from flutter app through JSON webToken (describe the process)
* Use web Token, use key on server 🡪 see in the literature, how to learn

## Hardware

* Arduino ESP8266
* NEO-6M / NEO-7M GPS module
* Temperature and Humidity Sensor module
* Insulated container
* App: tested in android but possible in ios

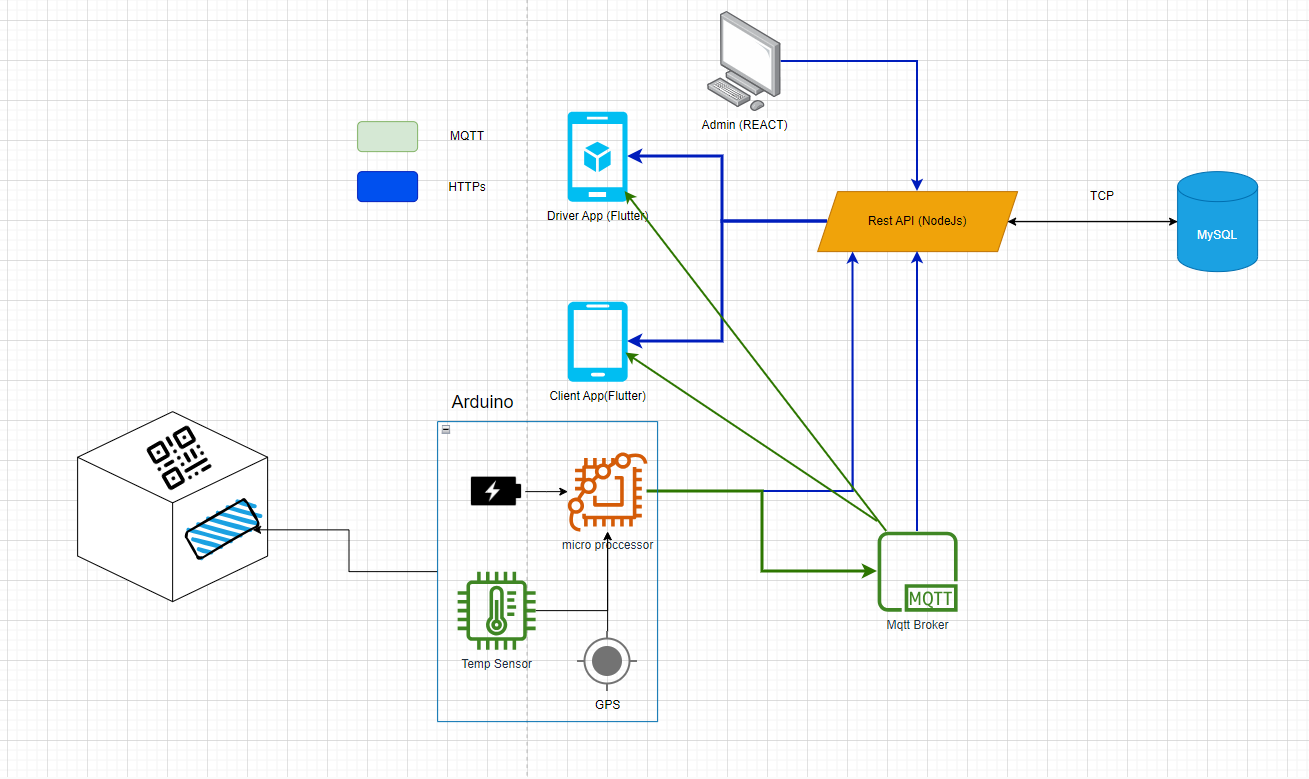
## Recommended software development environment

* Sense Layer: Arduino IDE
* Transportation Layer: HiveMQ for MQTT tests
* App Layer / Front End
  + Flutter (Mobile apps) 🡪 Visual Studio Code or Android Studio
    - Libs: ..
  + React for admin (web-based application) 🡪 Visual Studio Code
* Backend
  + Node.Js 🡪 Visual Studio Code or Android Studio
    - Libs:
  + DateBase 🡪 mySql

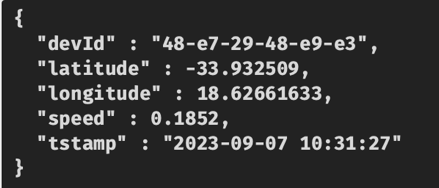
# Design

## Architecture

Proposed architecture – Compontent Overview



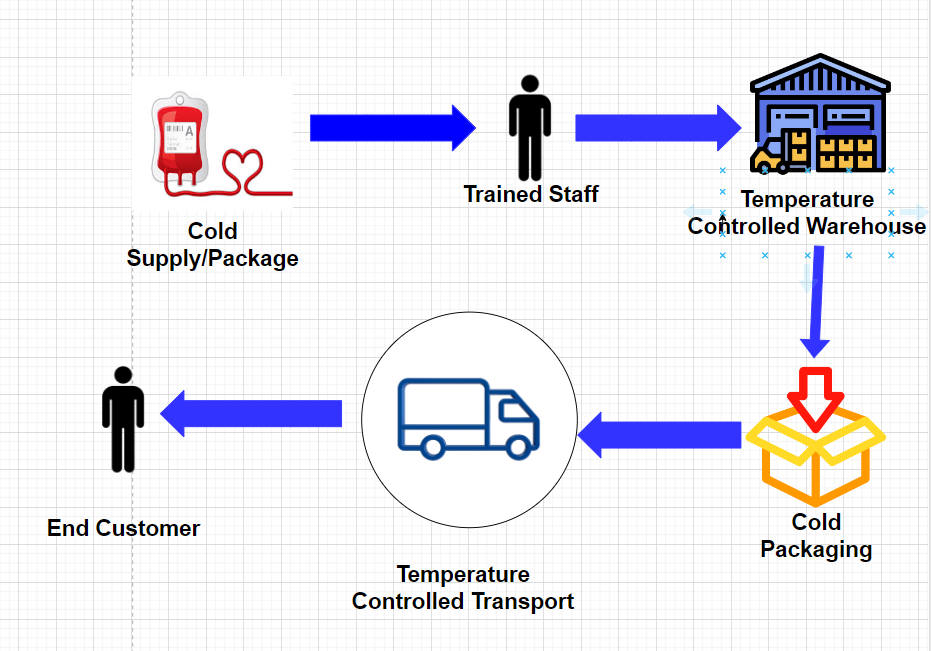
## Data Model for json-Object for IoTContainer



Data Model for DB

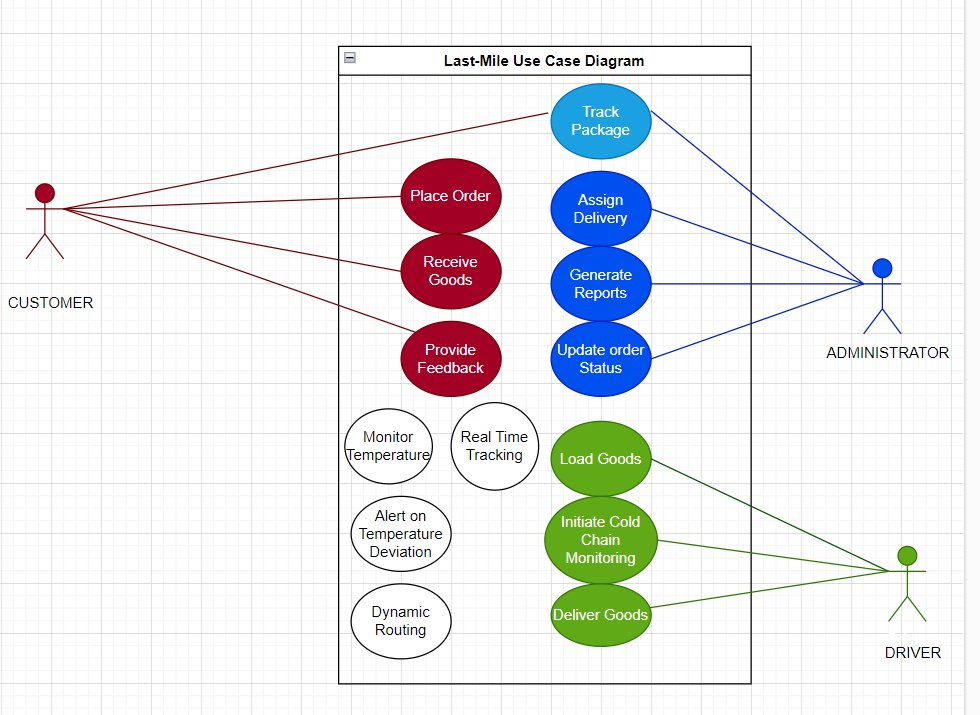
ER Diagram

## Process Model / Activity Diagrams



More in detail 🡪 activity diagram

Use case Diagram.



## Wireframes

# Implementation

Documentation REST-Api for node.js

How you use the TomTom-Api

## Smart Container

Github repository: https://github.com/dany-meyer/uwc\_tests/tree/main/Smart\_Container\_Arduino

## Driver App

Github repository: https://github.com/CS-UWC/Last-Mile/tree/master/driver\_cargo

## Admin App

Github repository: https://github.com/CS-UWC/Last-Mile/tree/master/cargo\_admin

## Dashboard

React

# Test

## Modul Test

**NEO-6M/7M GPS Module**

The GPS module is rather inconsistent in picking the satellite signals to generate the coordinates. Often time the GPS module must be exposed to the sky to get satellite signals this may be a suitable solution in highly flexible testing environments but becomes an issue when the module is closed inside the container.

The power supply to the module does not have many inconsistencies but does require a stable power supply as it has its own battery.

**Temperature/Humidity**  
The above modules only require a steady connection to the correct pins do not have any other inconsistencies associated with them.

## Integration Test

## Final Acceptance Test

Screen Shots

## Sources to learn from the Project

1. Already known
   1. Create Nodejs 🡪 tutorial ok
   2. Programming the micro controller 🡪 tutorial ok
   3. Create Flutter App 🡪 tutorial ok
   4. mySQL DB
2. Create Security with node.js using web Token 🡪 web link
3. Program Database using js 🡪 web link
4. Flutter and web Token 🡪 web link
5. Private cluster for MQTT
6. Learn the basics in react 🡪 web link
7. TomTom-API