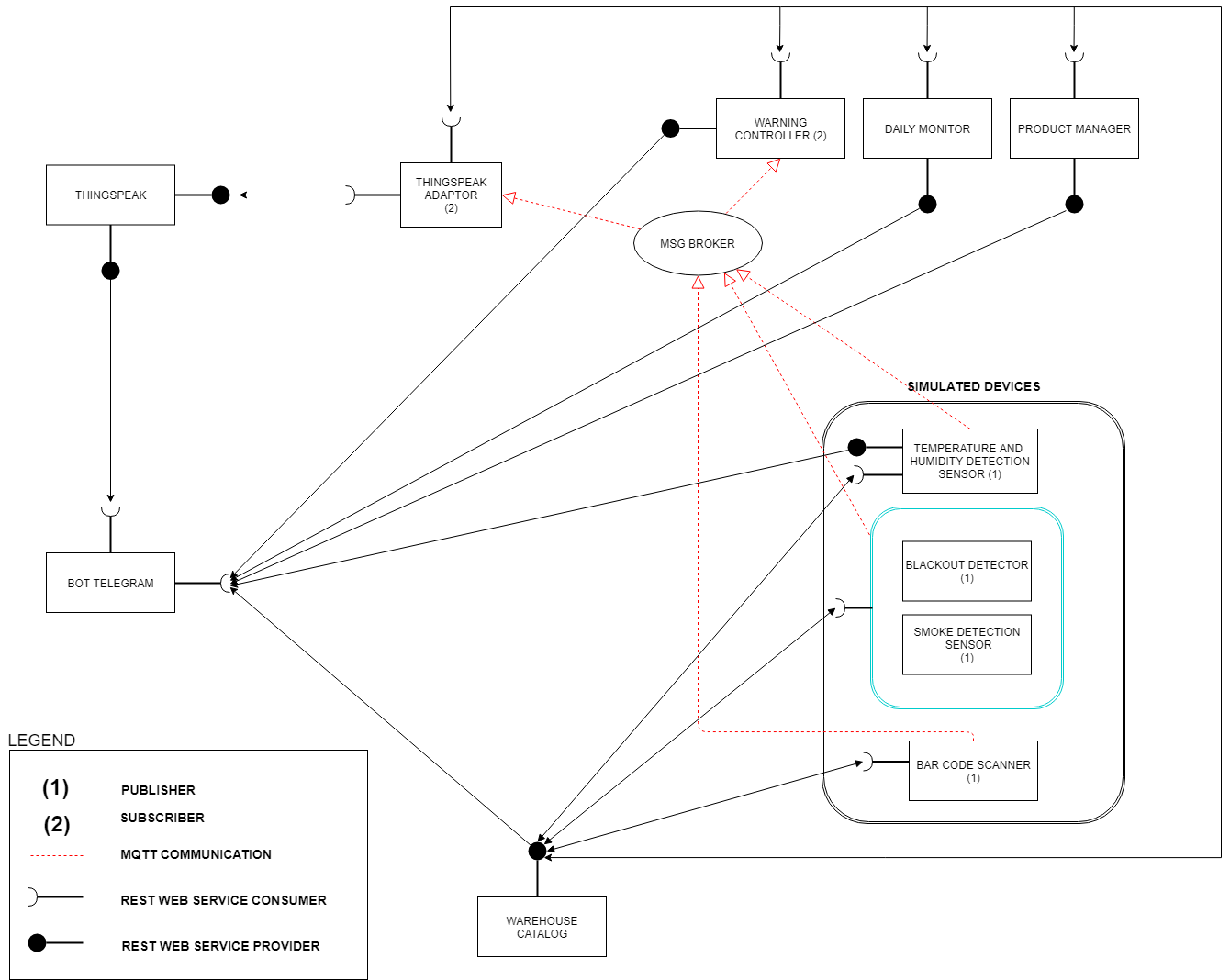
1. Name of Use Case

|  |  |
| --- | --- |
| **Name of the Use Case** | **Store Manager** |
| **Version No.** | v 0.2 |
| **Submission Date** | 09-12-2020 |
| **Team Members (with student ids)** | * Lazzarini Ilaria s277469 * Lia Caterina s277452 * Nicolicchia Riccardo s282357 * Vigliotti Manuela s287728 |

1. Scope and Objectives of Function

|  |  |
| --- | --- |
| **Scope and Objectives of Use Case** | |
| **Scope** | The proposed IoT platform provides services for smart warehouse management. |
| **Objective(s)** | Expected results are:  - providing a smart control of storage condition  - providing statistics on permanence of the products  - minimizing the resource waste. |
| **Domain(s)** | Smart Warehouse, Smart Building. |
| **Stakeholder(s)** | Warehouse workers, Warehouse owners. |
| **Short description** | The proposed IoT platform provides services for smart warehouse. The purpose of the IoT platform is to achieve proper control of the storage conditions of the products, in terms of temperature and humidity; it provides alerts when it detects smoke or current blackout. In addition, it allows an easier management in adding and removing products from the warehouse.  In summary, the main features offered are:   * remote control of the environmental conditions of the warehouse * control of the permanence of the products * statistics * system of alerts for critical events (blackout, smoke presence, too high/low temperature or humidity) * unified interfaces (i.e. REST Web Services and MQTT queues) available to enable Demand / Response. |

1. Diagram of Use Case



1. Complete description of the system

The proposed IoT platform for Smart Warehouse follows the microservices designing pattern. It also exploits two communication paradigms:

* publish/subscribe based on MQTT protocol and
* request/response based on REST Web Services.

In this context, twelve actors have been identified and introduced in the following:

* The **Message Broker** provides an asynchronous communication based on the publish/subscribe approach. It exploits the MQTT protocol.
* The **Warehouse Catalog** works as service and device registry system for all the actors. It provides information about end-points (i.e. REST Web Services and MQTT topics) of all the devices, resources and services in the platform. It also provides configuration settings for applications and control strategies (e.g. list of sensors). Each actor, during its start-up, must retrieve such information from the Warehouse Catalog exploiting its REST Web Services.
* The **Thingspeak Adaptor** is an MQTT subscriber that receives measurements on environmental measurements and upload them on Thingspeak through REST Web Services.
* **Thingspeak** is a third-party software (https://thingspeak.com/) that provides REST Web Services. It is an open-data platform for the Internet of Things to store, post-process and visualize data (through plots).
* **Telegram Bot** is a service to integrate the proposed infrastructure into Telegram platform, which is cloud-based instant messaging infrastructure. It retrieves measurements from simulated IoT devices, alerts, sensor measurement statistics and information about product management exploiting the REST Web Services.
* **Temperature and humidity detection sensors** work as publisher in the MQTT protocol: they send their measurement to the Message Broker at each fixed time. Then the message broker provides to send these data to the subscriber (warning controller). Data can be directly retrieved by Telegram Bot through REST Web Services.
* The **bar code scanner** is used to retrieve information about the products. It sends data to the Warehouse Catalog through REST Web Services, and it acts as a publisher in the MQTT protocol. It sends its data to the Message Broker every time new products are registered or removed. Produced data are analized by product manager which retrieves these data from warehouse catalog and sends product information to Telegram Bot through REST Web Services when user asks for it.
* The **blackout detector** and the **smoke detector** are alarm systems that send information about the criticality of the warehouse conditions. The blackout detector sends information at regular intervals, if no information is sent for a longer time a notification of power failure is sent. The smoke detector sends information about air conditions. Critical condition alarms are sent to Telegram Bot by warning controller through REST Web Services.
* The **warning** **controller** first acts as a subscriber receiving data with a specific topic from the Message Broker. Second, it provides a "control function". It processes this data for the purpose of detecting anomalies as too high/low temperature or humidity, smoke's presence or no current. In case of emergency some messages are sent to Telegram Bot by warning controller through REST Web Services.
* The **daily monitor** retrieves temperature and humidity data from the warehouse catalog through REST Web Services and it provides statistical values as mean, max and min with the timestamp of last sensors update. It evaluates data statistics considering the measurements from the first publication of the current day until the latest publication available. Data are sent to Telegram Bot through REST Web Services when user asks for it.
* The **product manager** retrieves codes from bar code scanner stored in the warehouse catalog through REST Web Services and it provides the amount of entry/exit products during the day and what kind of products are stored and their quantity. It controls the status of each product (entry or exit), the number that represents the quantity and the code which represents the product type. These data are sent to Telegram Bot through REST Web Services when user asks for it.