UNIVERSITÀ
DEGLI STUDI
DE L'AQUILA

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica

Università degli Studi dell'Aquila

Waterworks Autonomous System

2021 / 2022

Corso : Software Engineering for Autonomous System

Docente : *Davide Di Ruscio*

Project GitHub Repository: https://github.com/federix98/SE4AS

Progetto realizzato da:

Valerio Crescia	Matr. ***278270***	valerio.crescia@student.univaq.it
Federico Di Menna	Matr. ***253962***	federico.dimenna@student.univaq.it

UNIVERSITÀ DEGLI STUDI DE L'AQUILA

Requirements

Functional Requirements

1 - Scalare la portata d'acqua di una zona cittadina relativamente alla sua richiesta.

Priority: **Medium**

Il sistema dovrà adattare il flusso idrico delle diverse zone in maniera coerente con la loro richiesta.

2 - Identificazione anomalie

Priority: **High**

Marcare una zona come "in manutenzione" in caso di identificazione anomalie in modo da avere una perdita d'acqua minima.

3 - Il sistema può scalare la portata d'acqua delle zone in maniera proattiva invece che reattiva.

Priority: **Low**

Riguardo questo obiettivo è possibile pensare ad un approccio proattivo nel quale il sistema apprende continuamente le abitudini delle abitazioni della zona cittadina e scala la portata d'acqua in relazione alle predizioni di consumo che farà.

4 - Gestione dinamica delle zone

Priority: **High**

Il sistema sarà in grado autonomamente di riconoscere nuove zone da gestire e ignorare zone non più attive.

Non Functional Requirements

- **High Performance Database** (es. Time Series)
    - The system is supposed to contain a large amount of data, and it should be able to perform aggregation functions and real time queries on them, so we need a database which is designed for these purposes.
- **Scalability**
- **Reusability**
- **Cross-platform system**

COMPONENT DESCRIPTION

Managed System

Il sistema va a gestire un insieme di zone idriche, ogni zona è composta da un serbatoio il quale invia al sistema le seguenti informazioni:

- ### Livello di riempimento del serbatoio (Tank Fill Level)

  Percentuale che ci rappresenta il livello dell'acqua dentro il serbatoio.

- ### Flusso d'acqua in uscita dal serbatoio (Tank Output)

  Quantità d'acqua misurata in litri al secondo in uscita dal serbatoio che defluisce alle case della zone.

- ### Flusso d'acqua in entrata dal serbatoio (Tank Input)

  Quantità d'acqua misurata in litri al secondo in entrata dal serbatoio proveniente dalla condotta principale che serve tutte le zone.

- ### Indicazione se la zona è in manutenzione (Active)

  Valore booleano che ci indica se la zone è posta in manutenzione.

- ### Richiesta d'acqua della zona (Total Demand)

  Stima proveniente da tutte le abitazioni che ci fornisce un valore minimo misurato in litri al secondo della quantità d'acqua richiesta nella zona.

- ### Numero di abitazioni della zona (Num House)

  Intero che va a rappresentare il numero di abitazioni per zona.

- ### Numero di metri quadri della zona (Square Meters)

  Intero che va a rappresentare il numero di metri quadri contenuti per zona.

Oltre a queste classiche zone avremo che una zona speciale detta "ZONA0", la quale rappresenta la portante d'acqua che fornisce a tutti i serbatoi presenti nelle zone gestite. Essa ha come dato solo il flusso d'acqua, misurato in metri cubi al secondo, della condotta portante.

MAPE-K Loop Implementation

Architecture Diagram

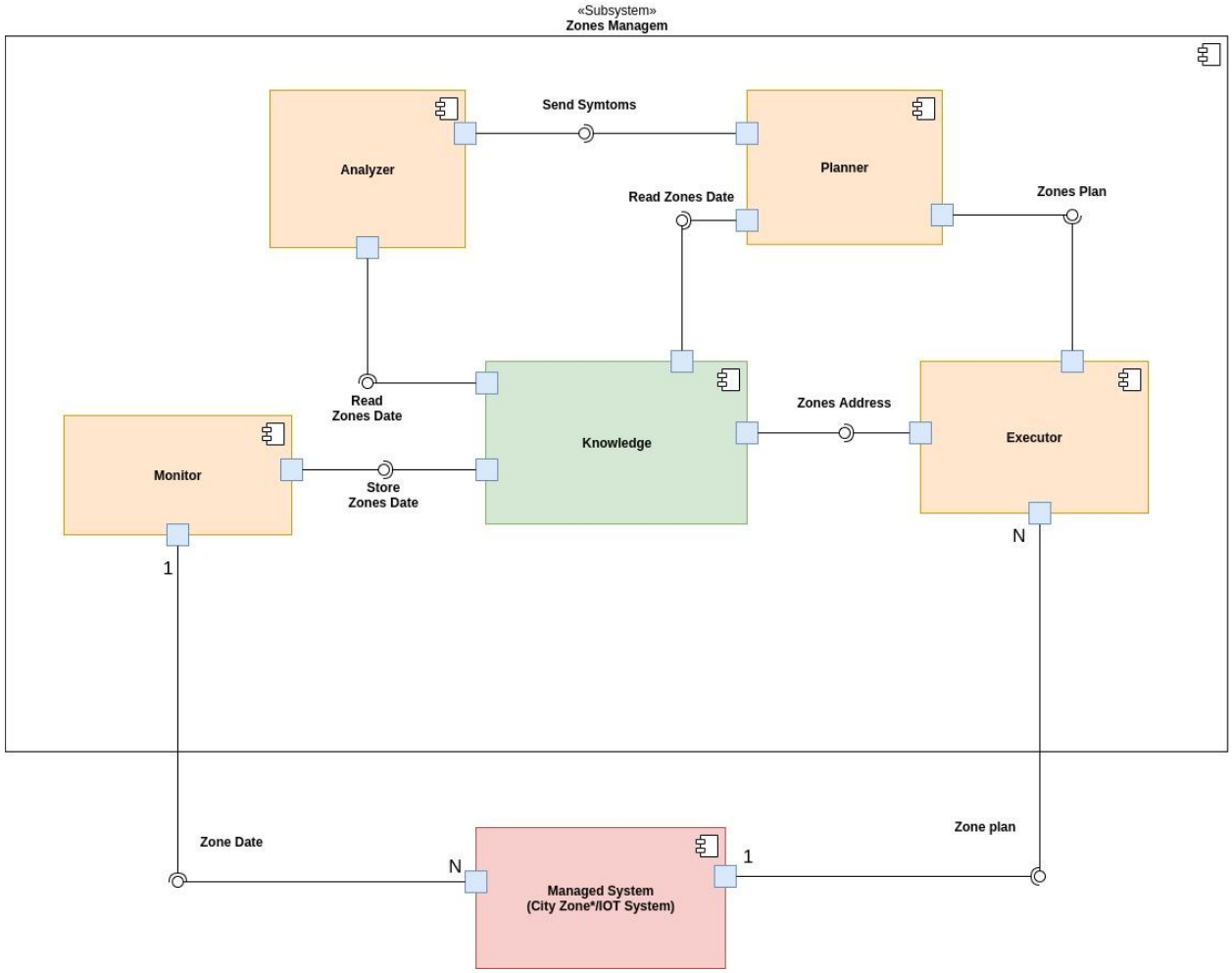

Figure 1: Water Autonomous System - MAPE-k Architecture

Pattern

In questo caso si è adottato il classico MAPE-K loop centralizzato perché il sistema non aveva requisiti di decentralizzazione e la pianificazione dell'adattamento va effettuata a livello globale per massimizzare l'adattamento alle regole impostate.

Monitor

Il componente monitor nel nostro sistema ha il compito di prendere tutti i dati mandati dalle zone e successivamente salvarli nella Knowledge.

Analyzer

L'Analyzer, nel nostro sistema, osserva tutti i dati salvati sul database dalle varie zone ed ad intervalli di tempo regolari, cattura i sintomi provenienti dalle zone.

Planner

Il Planner presente nel nostro sistema dovrà ricevere i vari sintomi dall'Analyzer, successivamente dovrà accedere alla knowledge per acquisire i dati correnti delle zone interessate e adatta i valori corrispondenti delle zone. Alla fine dell'adattamento dovrà procedere all'invio del piano all'executor.

Executor

L'Executor, nel nostro caso, dovrà ricevere il piano di adattamento dal planner e provvedere all'invio dei messaggi di adattamento dei parametri alle diverse zone.

Knowledge

La Knowledge dovrà contenere tutti i parametri delle varie zone, il riferimento alla zona che ha inviato il dato (topic e id) e l'esatto orario in cui è avvenuto l'inserimento dei dati. Siccome si ha un frequenza di inserimento dati molto elevata si avrà bisogno di un componente software in grado soddisfare i requisiti di performance richiesti.

SYSTEM ARCHITECTURE

Component Diagram

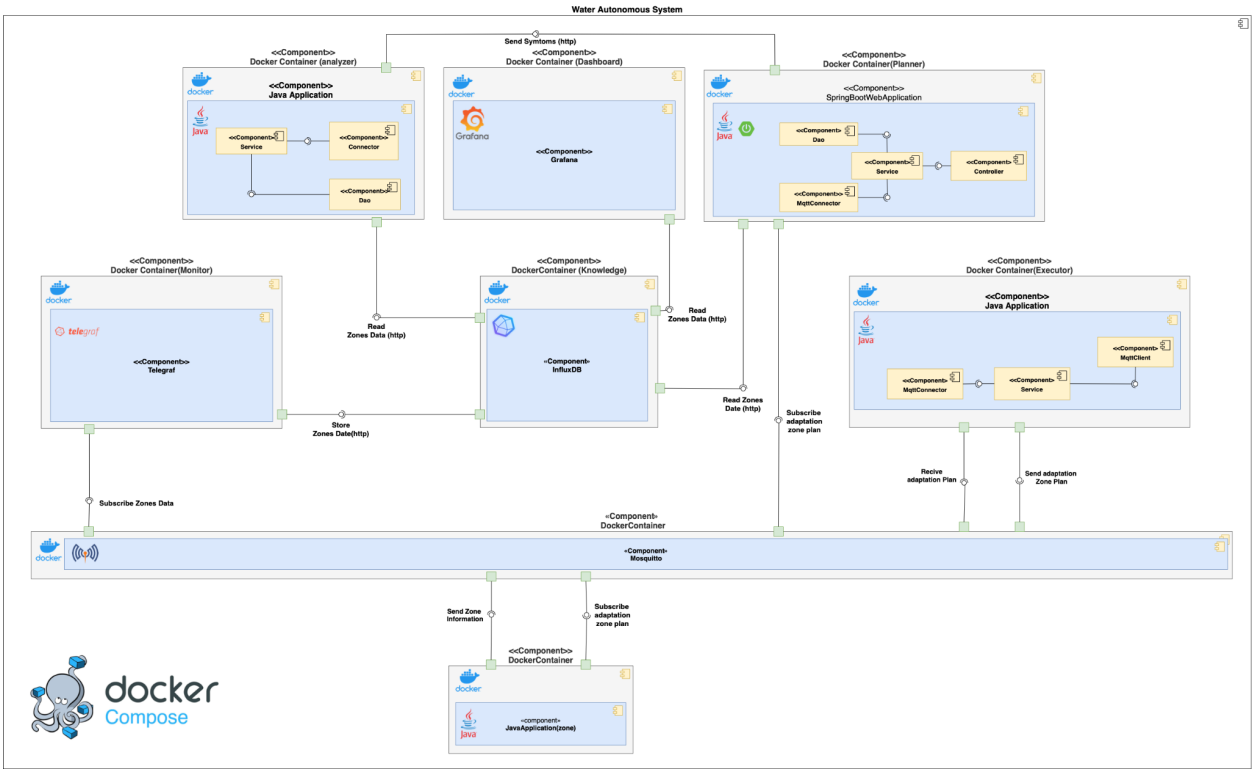

Figure 2: Water Autonomous System - Component Diagram

Per formare i componenti del nostro loop si è optato, per avere il massimo decoupling tra i vari componenti, un sistema a container. I componenti sono deployati su diversi container che comunicano attraverso l'esposizione delle porte interessate.

Knowledge

Per la knowledge si è optato per il database Influxdb il quale è di tipo TimeSeries e riesce a soddisfare i nostri requisiti di performance ed di modellazione dei dati provenienti dal componente monitor.

Monitor

Il componente monitor viene rappresentato nel nostro sistema dal container Docker che racchiude al suo interno il plugin Telegraf il quale osserva le varie comunicazioni mqtt inviate dalle zone, ne prende i vari dati e successivamente li salva nella knowledge. La scelta di questo componente deriva dalla perfetta simbiosi con il database Influxdb.

Analyzer

Per il componente Analyzer si è optato per una semplice Java application che esegue in maniera temporizzata un thread. Esso effettua query sul database e se rivela sintomi su di esso provvede a comunicarlo al componente Planner inviandogli il riferimento del sintomo trovato tramite una connessione Http ad un preciso endpoint.

Planner

Il componente Planner è rappresentato con il Docker container che racchiude un applicativo SpringBoot. Esso riceve dal Analyzer i vari sintomi tramite connessione Http ed esegue un adattamento andando prima ad interrogate la Knowledge e successivamente applicando la policy. L'adattamento viene convertito in un insieme di direttive che saranno spedite all'Executor.

Executor

L'executor è una Java Application che riceve un singolo adattamento per l'intero sistema. Esso analizza l'adattamento globale e successivamente manda i singoli adattamenti a ogni zona.

MQTT Broker

Per eseguire le connessioni MQTT si è utilizzato il broker mosquitto racchiuso in Docker container.

Dashboard

Per osservare i vari adattamenti del sistema si è utilizzata la dashboard Grafana collegata direttamente alla Knowledge.

Sequence Diagrams

Per illustrare il flusso dati del sistema, abbiamo preso come esempio il requisito funzionale #1. I seguenti sequence diagram ne descrivono i dinamismi.

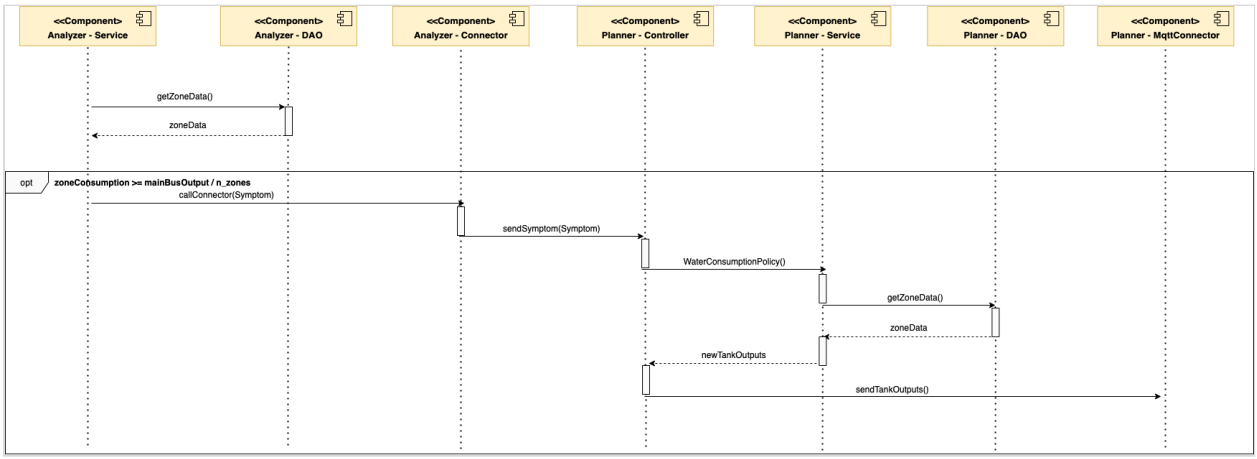

Figure 3: Analyzer - Planner interaction flow

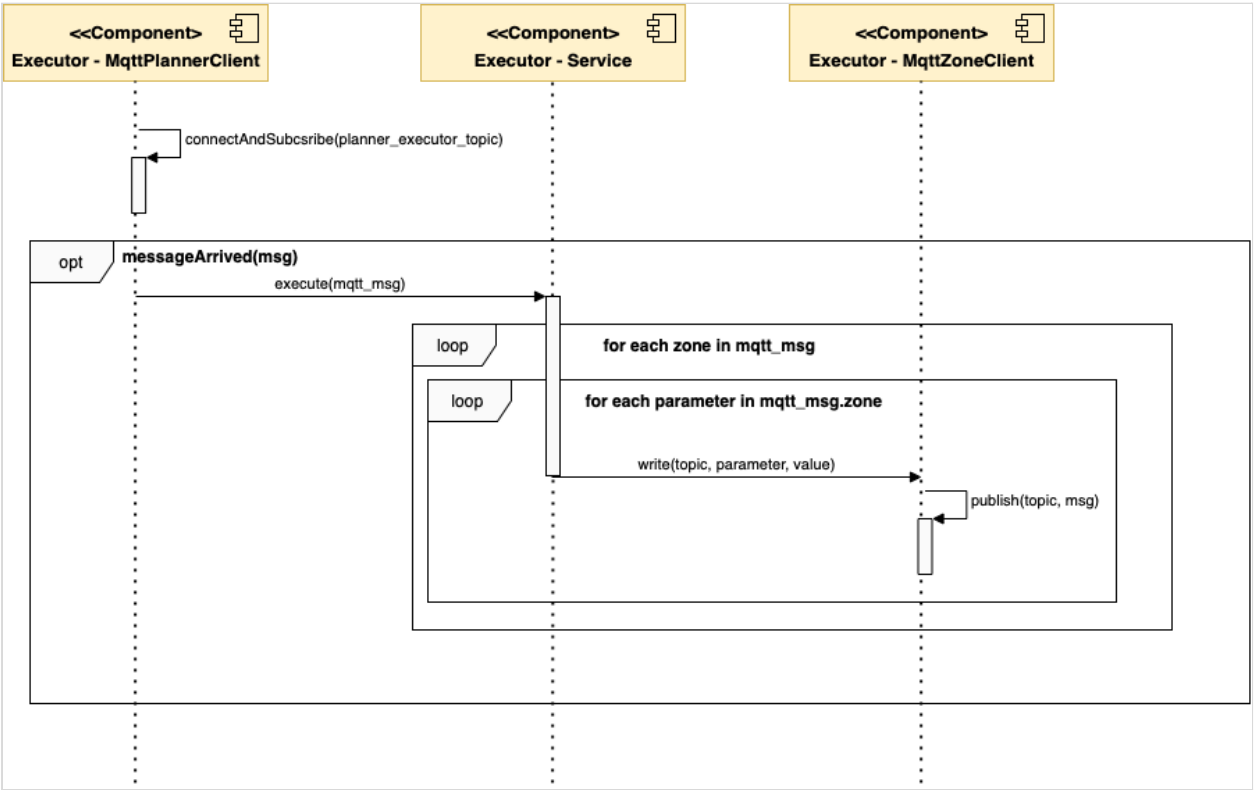

Figure 4: Executor flow

Nella figura 3 viene mostrato il flusso di operazioni compiuto dai componenti Analyzer e Planner per gestire il sintomo "Consumption Adaptation". L'entry point del diagramma è l'analyzer, il quale è stato schedulato per eseguire l'analisi periodicamente. Ogni qualvolta viene effettuata l'analisi, inizialmente il componente Analyzer - Service recupera gli ultimi dati delle varie zone tramite il componente Analyzer - DAO.

Successivamente, viene eseguito il seguente check per ogni zona:

```
if (zone.getDemand() >= totalOutput / number_of_zones)
```

In caso il check risulti vero, verrà generato un sintomo di tipo "Consumption Adaptation" che verrà inviato al componente Analyzer - Connector, il quale comporrà e invierà il corrispondente messaggio json verso il planner. Quest'ultimo, ricevuto il messaggio dall'analyzer, invoca la Water Consumption Policy dichiarata al suo interno che scalerà gli output delle zone per adattarli alla loro richiesta. Per fare ciò, avrà bisogno di contattare il componente Planner - DAO per recuperare i dati delle varie zone. Una volta calcolati i valori di output, essi verranno inviati sul broker mqtt mediante il componente Mqtt Connector.

A questo punto, l'executor che possiede un client mqtt che precedentemente ha eseguito il subscribe() sullo stesso topic, riceverà il messaggio inviato dal planner. Il messaggio contiene i dati aggiornati di ogni zona che il componente Executor - Service è in grado di inoltrare sui rispettivi topic di acting delle zone. Per fare ciò, è necessaria una prima iterazione sulle zone ricevute dal planner, e una seconda iterazione annidata su tutti i parametri da aggiornare.

ARCHITECTURAL TESTING / IMPROVING

Architettura corrente

Dopo aver testato il sistema utilizzando gli attuali componenti, abbiamo voluto sostituire uno dei componenti del loop per verificare l'efficacia dell'architettura in termini di decoupling e integrazione. Il componente che è stato sostituito è il planner, che passa da un approccio di tipo reattivo a uno proattivo.

Nuovi componenti

Il nuovo planner sarà un nuovo progetto Java integrato in termini di connettori con gli altri, che adotterà un approccio di tipo proattivo, quindi le policy faranno uso di tutti i dati presenti nella knowledge piuttosto che utilizzare soltanto le osservazioni correnti (approccio reattivo). In questo modo il sistema utilizzerà tutti i dati presenti nella knowledge, ma eventuali variazioni immediate a breve termine non verrebbero soddisfatte.

Approccio Proattivo - Predictor Description

Per adottare un approccio proattivo alla pianificazione delle azioni da intraprendere per effettuare l'adattamento, abbiamo pensato di utilizzare un **regressore lineare** che proietta il prossimo dato sulla base degli ultimi *n* analizzati. In particolare, viene estrapolata una media ogni 10 minuti dei valori della total demand della zona e viene realizzato il regressore.

La query per estrapolare i dati aggregati per 10 minuti (ad esempio in zona 1) è la seguente:

```
SELECT mean(totalDemand) FROM zone WHERE topic = 'home/sensing/zone1' GROUP BY time(10m)
```

Facilmente realizzabile e editabile utilizzando le funzioni di aggregazione di influxdb.

Estratta la serie di valori dalla query precedente, il sistema effettua una predizione sul valore successivo della serie utilizzando una semplice regressione lineare in cui la variabile indipendente X riferisce al tempo (intervallo t - n / t_0) e la variabile dipendente Y riferisce alla domanda. La formula utilizzata è la seguente:

$$a = \frac{(\Sigma y)(\Sigma x^2) - (\Sigma x)(\Sigma xy)}{n(\Sigma x^2) - (\Sigma x)^2}$$
$$b = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{n(\Sigma x^2) - (\Sigma x)^2}$$

Figure 8: Linear Regression Equation

Utilizzando questo semplice metodo possiamo eventualmente predire un valore medio in un intervallo di tempo successivo alla serie attuale sfruttando il trend attuale della demand e quindi capire se costante, ascendente o discendente.

Facendo un esempio pratico prendiamo la seguente serie di consumi medi per 10 minuti: $\{85, 83, 84, 82, 81, 77, 79, 76\}$, il regressore calcolerà l'equazione della retta mostrata in figura seguente in modo da prevedere l'andamento del prossimo punto ($y = -1.25x + 85.25$). Il risultato della predizione su un potenziale nono valore sarà 75.25.

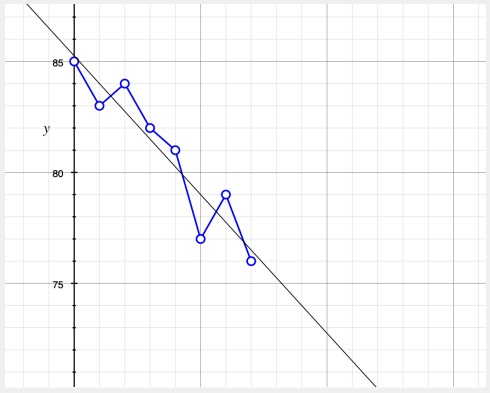

Figure 9: Linear Regression - real example usage

Results

La figura 9 rappresenta i dati adattati passando da un approccio reattivo ad uno proattivo. In particolare, nella sezione centrale "tank output" è possibile osservare come il valore di tank output sia discontinuo nell'approccio reattivo (prima parte della curva) al contrario della continuità della parte finale della curva (approccio proattivo).

How FR have been addressed

Functional Requirement #1

Il requisito funzionale #1 è stato raggiunto aggiungendo un particolare sintomo chiamato "consumptionAdaptation" grazie al quale il sistema è in grado di accorgersi quando una zona sta consumando più del previsto e adattare l'output dei serbatoi in base alla richiesta. Nella figura 5 viene mostrato come il sistema adatta il Tank Output in base alla Water Demand delle zone.

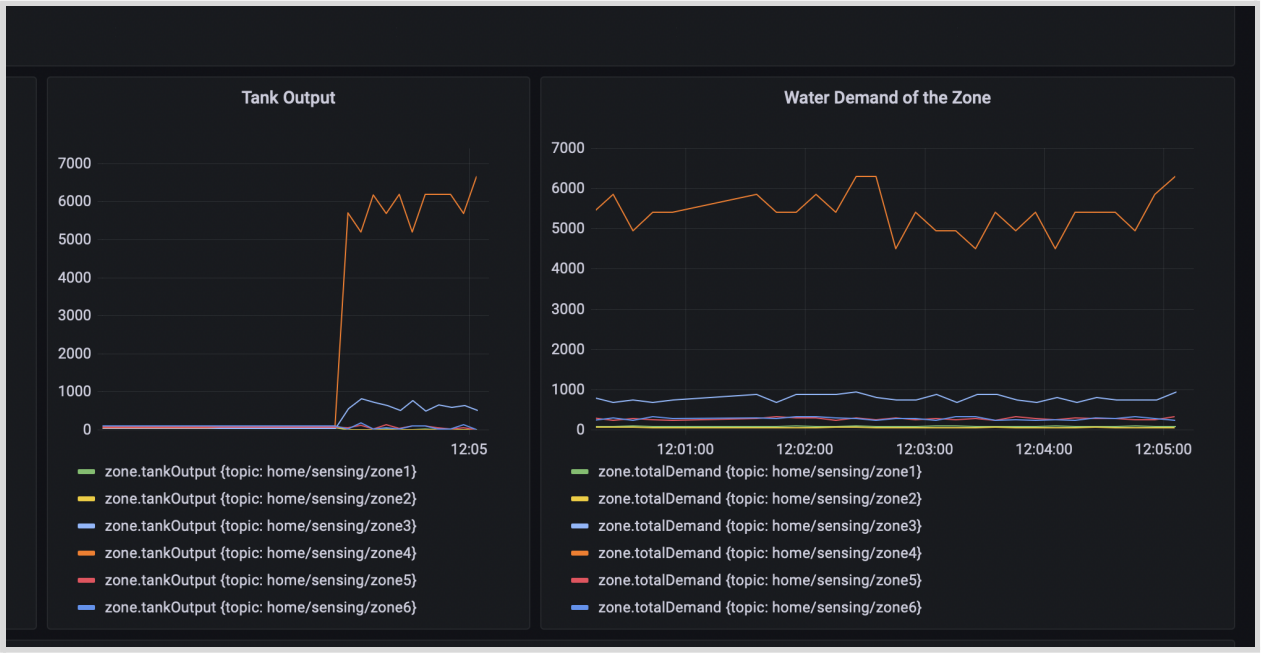

Figure 5: Grafana Consumption Adaptation Panels

Functional Requirement #2

Il requisito #2 è stato raggiunto utilizzando il sintomo "MAINTENANCE" che viene scatenato dal planner quando determinati parametri sono totalmente fuori la norma. In particolare un consumo eccessivo d'acqua da parte di una zona (consumo >= numero di abitazioni * 30) e quando il serbatoio si svuota. La figura 6 mostra le sezioni della Grafana Dashboard dedicata al controllo della manutenzione delle zone. Nella figura 7 viene mostrato il livello di riempimento delle zone. Da come possiamo osservare le zone con il serbatoio vuoto sono state messe in manutenzione (OFF in figura 6).

Una zona in manutenzione può tornare in attività modificando manualmente (simulazione di un operatore) il flag active da 0 a 1. In questo modo il sistema verificherà se le condizioni critiche sono

state realmente rimosse. Nel caso in cui non ci siano condizioni critiche il sistema considererà la zona normalmente, in caso contrario metterà nuovamente la zona in manutenzione.

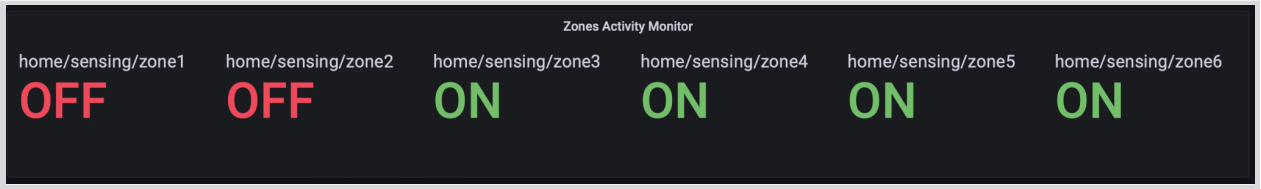

Figure 6: Grafana Maintainance Panels

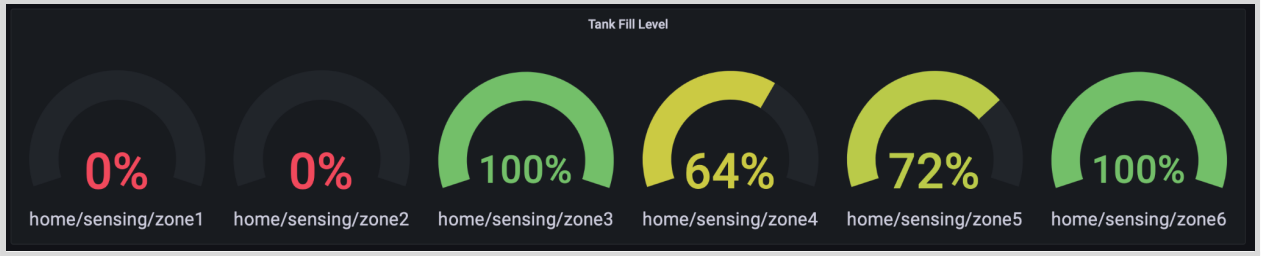

Figure 7: Grafana - Livello riempimento serbatoi

Functional Requirement #3

Questo requisito è stato gestito tramite la sostituzione del componente planner descritta nel paragrafo precedente.

Functional Requirement #4

Il requisito #4 può essere considerato come trasversale rispetto agli altri dato che non riguarda un particolare sintomo, ma su quali zone deve agire il sistema autonomo. A tale scopo è stato necessario gestire due casi fondamentali:

- Aggiunta dinamica della zona: Gestire questo caso non è stato complesso considerando il come è stato progettato il sistema. Dato che il monitor è sottoscritto ad ogni topic sotto uno specifico path (home/sensing), per ogni nuova zona basterà inviare il messaggio sul topic /home/sensing/zoneID e il sistema andrà a considerare in maniera del tutto autonoma la nuova zona.

  ASSUNZIONE: Ogni zona possiede un ID identificativo univoco rispetto alle altre. Il sistema non si occupa di gestire l'assegnamento degli ID in modo dinamico.

- Rimozione dinamica di una zona che viene più monitorata: Per gestire questa problematica è stato scelto di basarsi su una specifica finestra temporale. In questo modo tutti i messaggi inviati prima della finestra scelta non vengono considerati e quindi la zona non verrà più considerata (in questo modo una zona può anche tornare in attività in ogni momento).

  Attualmente la finestra temporale è settata a 7 giorni

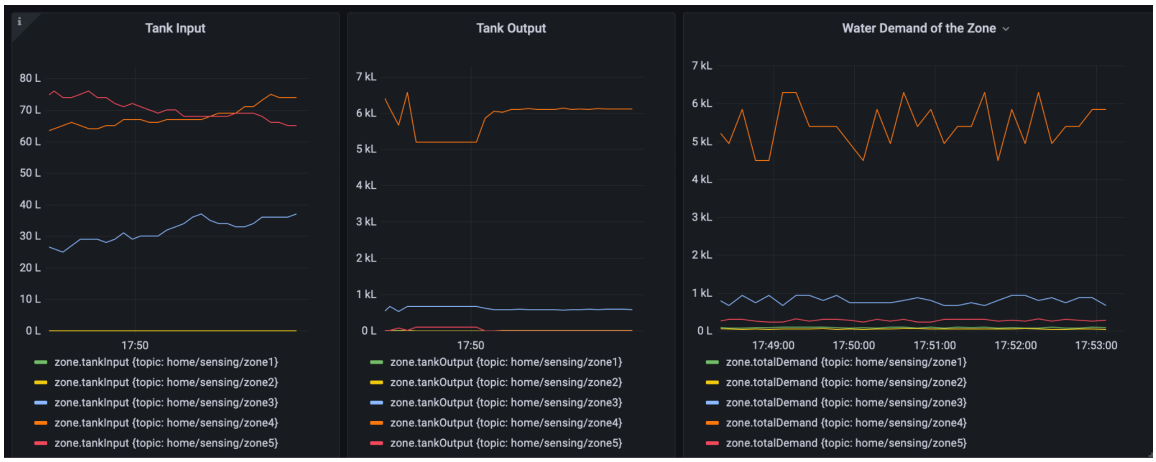

Figure 9: Proactive Planner results

TECHNOLOGIES