Programare Orientată pe Obiecte

Tema2

Termostat Inteligent

Responsabili temă: Marilena Panaite
Deadline: 3.12.2019 (Deadline hard: 10.12.2019)

Introducere

Scopul acestei temei este de a optimiza încălzirea unei locuințe (poate fi apartament sau casa) prin analiza detaliată a senzorilor de temperatură. În vederea realizării analizei, au fost montati senzori de temperatură în fiecare cameră din casa. Astfel, fiecare dispozitiv va capta timpul și valorile senzorului și va transmite informațiile către un sistem central pentru analiză și stabilirea trendurilor de temperatura pentru anumite intervale de timp și pentru a porni sistemul central de încălzire (centrala termica).

Descrierea problemei

Prima etapă constă în analiza configurației casei alese și montarea de senzori de temperatură în fiecare dintre camere. Apoi, fiecare senzor va fi configurat să transmită informații cu valorile captate la anumite intervale de timp. De asemenea, proprietarul casei va preciza ce temperatura optima își dorește pentru întreaga casa. În tema aceste informații vor fi prezente în prima parte a fișierului "*therm.in"*.

A doua etapă a analizei constă în colectarea datelor de la dispozitivele IoT instalate pe o perioadă de timp și stocarea datelor într-un mod optimizat pentru a facilita analiza trendurilor de temperatură și evoluția lor. Avand in vedere că putem să transmitem observații de la senzori la intervale mici de timp, putem colecta date substanțiale într-o perioada scurta de timp.

Pentru a stoca într-un mod eficient toate aceste date despre temperatura, va fi folosit conceptul de Time Series, unde vom stoca datele despre temperatura pentru fiecare camera în parte mapand timpul primit ca parametru într-un unul din intervale și stocand valorile din acea ora în acel interval:

1. Lungimea unui interval va fi de 1h și va conține valorile transmise în aceea ora
2. Pentru simplitate și testabilitate, timpul de referinta va fi dat ca parametru. Toate valorile transmise de la senzori vor avea un timestamp mai mic decât cel de referinta.
3. Pentru un senzor putem primi și putem stoca pana la 24h în urma valorile temperaturilor
4. Temperaturile dintr-un interval vor fi stocate în ordine crescătoare a valorilor lor
5. Timpul primit ca parametru al observatiei va fi Unix Timestamp in secunde

A treia etapă va consta în calcularea trendurilor pentru fiecare cameră în parte şi apoi pentru întreaga locuinţă pentru a stabili daca centrala termică va trebui pornita.

Pentru tema actuală, algoritmul pornire a centralei (cand este intalnita comanda TRIGGER HEAT) se va calcula astfel:

1. Se vor calcula pentru fiecare camera din casa temperatura cea mai mica din ultima ora.
2. Temperatura medie a casei va fi apoi calculată ca media ponderată între **temperaturile minime** din fiecare camera din **ultima ora** în funcţie de **suprafaţa** camerelor.

De asemenea, de-a lungul transmiterii observaţiilor şi consultarii termostatului inteligent, proprietarul poate schimba temperatura globală dorita a casei prin comanda TEMPERATURE.

Bonus

Sistemul trebuie sa fie capabil sa capteze şi să stocheze informaţii despre nivelul de umiditate din locuinţă. Senzorul de umiditate va fi montat pe acelaşi dispozitiv IoT pe care a fost montat cel de temperatură. Acesta va trimite date către sistemul central în acelaşi mod în care trimite senzorul de temperatura.
Valoarea umiditatii va influenţa pornirea centralei astfel:

1. Pentru comanda TRIGGER HEAT, vom calcula media umiditatii în mod asemanator cu cea a temperaturii: media ponderată între **valorile maxime ale umiditatii** din fiecare camera din **ultima oră** în funcţie de **suprafaţa** camerelor.
2. Dacă media umiditatii este mai mare decat umiditatea dorita, atunci centrala nu va fi pornita indiferent de media temperaturilor.

Cerinte

1. Sa se implementeze stocarea datelor despre temperatura pentru o casa cu configuraţia camerelor date împreuna cu senzorii montati în fiecare camera.
    a. Se vor stoca datele pentru fiecare senzor în parte conform cerinţelor de mai sus (Time Series cu intervale de o ora)
    b. Se vor folosi doar structurile de date din *Java Collection Framework*
    c. Specificati în README motivele alegerii fiecărui tip din *Collection*
2. Sa se implementeze comenzile prezente in *therm.in*: OBSERVE, TRIGGER HEAT, TEMPERATURE şi LIST

Bonus

1. Sa se implementeze în plus stocarea umiditatii din fiecare camera în mod asemanator cu temperatura (folosind Time Series şi intervale de ora). Comanda de implementat din *therm.in* va fi OBSERVEH.

2. Sa se modifice implementarea comenzii TRIGGER HEAT astfel incat daca media umiditatii din ultima oră este mai mare decat umiditatea dorită nu va porni centrala indiferent de valoare temperaturii.

Testare

Tema va fi testată pe platforma *Vmchecker* cu ajutorul testelor automate. Este obligatoriu ca în arhiva temei sa aveţi un Makefile în care sa implementati directive pentru compilare şi rulare (folosind metoda *main*).

Datele vor fi citite din fişierul *therm.in* şi rezultatele vor fi scrise în therm.out.

Exemplu de fisier de intrare *therm.in* - va contine date pentru o singura casa:

therm.in	
3 23 1573776000	<nr_camere> <temperatura_globala> <timestamp_referinta> Numărul de camere din casa, temperatura dorită pentru casa, timpul de referinta pentru temperaturile transmise
ROOM1 345ASDJ 16	<nume_camera> <device_id> <suprafata> Nume camera, device_id pentru senzor temperatura şi suprafaţa camerei
ROOM2 425FDAK 20	
ROOM3 535PLKA 22	
OBSERVE 345ASDJ 1573706510 21.5	OBSERVE <device_id> <timestamp> <temperatura> <temperatura> - double Va insera valoarea temperaturii corespunzătoare device_id-ului în interval de timp asociat cu timestamp-ul dat.
TRIGGER HEAT	Calculeaza conform algoritmului din cerinţa nevoia de a porni centrala termica.

	Output: YES or NO
TEMPERATURE 22	TEMPERATURE <temperatura_globala> Schimba valoarea temperaturii dorite global. Orice comanda de TRIGGER HEAT va lua în considerare noua temperatura.
LIST ROOM1 1573706910 1573706910	LIST ROOM1 <start_interval> <stop_interval> Valorile temperaturilor observate între cele doua intervale de timp date ca parametru. Acestea vor fi afişate în ordinea crescătoare a temperaturii. Daca se incadreaza mai multe intervale, doar fiecare interval se va afisa sortat nu intreaga secventa. Output: ROOM1 21.5
BONUS	
3 23 25 1573776000	<nr_camere> <temperatura_globala> <umiditate_globala> <timestamp_referinta> Numărul de camere din casa, temperatura şi nivelul de umiditate dorite pentru casa
OBSERVEH 345ASDJ 1573706510 19	OBSERVEH <device_id> <timestamp> <umiditate> <umiditate> - poate fi *double* Va insera valoarea umiditatii corespunzătoare device_id-ului în interval de timp asociat cu timestamp-ul dat.

Punctaj

Punctajul pe tema se va imparti altfel:

- 4.5p Teste publice

- 4.5p Teste private
- 1p Javadoc
- 2p Bonus

Depunctari:

- -5p dacă nu sunt folosite structurile din Java Collection Framework pentru stocare (https://docs.oracle.com/javase/8/docs/technotes/guides/collections/reference.html)
- -2p dacă datele nu sunt stocate pe buckets de 1 ora
- -1p daca nu exista README

Referinte

[1] http://mathworld.wolfram.com/WeightedMean.html
[3] https://en.wikipedia.org/wiki/Time_series_database
[4] https://redislabs.com/redis-best-practices/time-series/sorted-set-time-series/
[5] https://docs.oracle.com/javase/8/docs/technotes/guides/collections/reference.html
[6] https://www.unixtimestamp.com/