

Tema2

Termostat Inteligent

Responsabili temă: Marilena Panaite

Deadline: 3.12.2019 (Deadline hard: 10.12.2019)

Introducere

Scopul acestei teme 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 montați 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 optimă 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. Având în vedere că putem să transmitem observații de la senzori la intervale mici de timp, putem colecta date substanțiale într-o perioadă scurtă 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 mapând timpul primit ca parametru într-unul din intervale și stocând 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 referință va fi dat ca parametru. Toate valorile transmise de la senzori vor avea un timestamp mai mic decât cel de referință.
3. Pentru un senzor putem primi și putem stoca până 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 observației va fi Unix Timestamp în secunde

A treia etapă va consta în calcularea trendurilor pentru fiecare cameră în parte și apoi pentru întreaga locuință pentru a stabili dacă centrala termică va trebui pornită.

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

1. Se vor calcula pentru fiecare cameră din casă temperatura cea mai mică din ultima oră.
2. Temperatura medie a casei va fi apoi calculată ca media ponderată între **temperaturile minime** din fiecare cameră din **ultima oră** în funcție de **suprafața** camerelor.

De asemenea, de-a lungul transmiterii observațiilor și consultării termostatlui inteligent, proprietarul poate schimba temperatura globală dorită a casei prin comanda TEMPERATURE.

Bonus

Sistemul trebuie să fie capabil să capteze și să stocheze informații despre nivelul de umiditate din locuință. Senzorul de umiditate va fi montat pe același dispozitiv IoT pe care a fost montat cel de temperatură. Acesta va trimite date către sistemul central în același mod în care trimite senzorul de temperatură.

Valoarea umidității va influența pornirea centralei astfel:

1. Pentru comanda TRIGGER HEAT, vom calcula media umidității în mod asemănător cu cea a temperaturii: media ponderată între **valorile maxime ale umidității** din fiecare cameră din **ultima oră** în funcție de **suprafața** camerelor.
2. Dacă media umidității este mai mare decât umiditatea dorită, atunci centrala nu va fi pornită indiferent de media temperaturilor.

Cerinte

1. Să se implementeze stocarea datelor despre temperatura pentru o casă cu configurația camerelor date împreună cu senzorii montați în fiecare cameră.
 - a. Se vor stoca datele pentru fiecare senzor în parte conform cerințelor de mai sus (Time Series cu intervale de o oră)
 - b. Se vor folosi doar structurile de date din *Java Collection Framework*
 - c. Specificați în README motivele alegerii fiecărui tip din *Collection*
2. Să se implementeze comenzile prezente în *therm.in*: OBSERVE, TRIGGER HEAT, TEMPERATURE și LIST

Bonus

1. Să se implementeze în plus stocarea umidității din fiecare cameră în mod asemănător cu temperatura (folosind Time Series și intervale de oră). 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 aveți un *Makefile* în care sa implementati directive pentru compilare și rulare (folosind metoda *main*).

Datele vor fi citite din fișierul *therm.in* și 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 și suprafața 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 cerința 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	LIST ROOM1 <start_interval> <stop_interval> Valorile temperaturilor observate între cele doua intervale de timp date ca parametru. Acestea vor fi afișate în ordinea crescătoare a temperaturii. Output: ROOM1 21.5
BONUS	
3 23 25	<nr_camere> <temperatura_globala> <umiditate_globala> Numărul de camere din casa, temperatura și nivelul de umiditate dorite pentru casa
OBSERVEH 345ASDJ 1573706510 19	OBSERVEH <device_id> <timestamp> <umiditate> <umiditate> - poate fi <i>double</i> 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 dacă nu există 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/>