Protocolli IioT - Test finale

Adv Networking:

comunicazione tra oggetti e standard di interoperabilità

Alunno : Vendrame Alessandro

**Testo del problema:**

Un’azienda che gestisce flotte di veicoli elettrici (automobili) ha necessità di monitorarla per conoscere in qualsiasi momento stato e posizione di ciascun veicolo per poter poi estrarre informazioni rilevanti dai comportamenti di guida dei conducenti.

Ogni automobile è dotata di una centralina intelligente (con prestazioni simili a un

Raspberry) e l’alimentazione data dalla batteria dell’auto ne garantisce una discreta

autonomia.

Le flotte di veicoli sono attive in 4 città (Milano, Roma, Napoli e Pordenone). Per ciascun

veicolo è necessario:

* leggere e memorizzare in un sistema su Cloud i tipici dati di telemetria (velocità, posizione, consumi, stato batteria, ecc.)
* poter inviare comandi da applicazioni mobili collegate a un sistema Cloud (apertura porta, accensione 4 frecce, apertura nuova corsa, chiusura corsa)

Ciascuna flotta ha circa un centinaio di veicoli. Essi possono essere Tesla model S oppure Fiat 500.

Il sistema Cloud avrà il compito di elaborare le informazioni ricevute:

* generando statistiche (km percorsi, consumo medio, ecc.) per ciascuna città
* generando statistiche (km percorsi, consumo medio, ecc.) per ciascun tipo di veicolo
* inviare a un sistema esterno (via chiamate API) tutti i dati di telemetria ricevuti dalle Tesla
* raccogliere in un database dedicato (es. di tipo Time Series) i soli dati raccolti per la città di Pordenone

**Compiti:**

1. Rappresentare su un diagramma l’architettura della soluzione: entità in gioco (su veicoli e su Cloud) e interconnessioni fra esse
2. Definire e motivare con quali protocolli ciascuna entità è interconnessa
3. Definire come sono strutturati i messaggi / chiamate per lo scambio di informazioni
4. Proporre una revisione (se necessario) all’architettura nel caso la centralina installata sul veicolo fosse meno performante (PIC anziché Raspberry) e con batteria autonoma, quindi di potenza / durata inferiore

Si prega di esporre i concetti in maniera chiara e ordinata. Testi o diagrammi difficilmente

leggibili o disordinati saranno considerati errati.

**Ragionamento**

Dopo aver analizzato il testo possiamo osservare che:

* I dati ricevuti dalla città di Pordenone devono essere salvati in un database separato
* I dati ricevuti dalle Tesla dovranno essere inviati ad un altro servizio tramite chiamate API
* Le auto potranno ricevere comandi dall’esterno inviati da applicazioni mobile
* I dati una volta salvati dovranno generare statistiche, per città o per modello dell’auto

Una volta ottenute queste prime informazioni possiamo quindi pensare di stilare uno schema dove andremo a mostrare come comunicano i dispositivi con il cloud.

*Macchina:*

La nostra macchina è provvista di una centralina intelligente che raccoglie i dati dai sensori presenti nell’auto e li invia ad un broker AMQP, in quanto offre la possibilità di poter separare i dati ricevuti in diverse code, in modo da poter gestire al meglio la suddivisione tra città.

Per fare in modo che i messaggi vengano salvati in caso di guasto del server o perdita di connessione, si è pensato di utilizzare il servizio offerto da Redis in locale, in quanto ci permetterà di utilizzare una coda dove verranno salvati i dati telemetrici e che appena ce ne sarà la possibilità li invii al server.

Non dimentichiamo però che la macchina dovrà anche poter ricevere dei comandi dall’esterno, sarà quindi necessario connettere la centralina ad un broker MQTT e fare il subscribe ad un determinato topic, che vedremo più tardi.

*Broker MQTT:*

Per la comunicazione tra il i dispositivi mobili e le macchine, si è pensato di utilizzare un broker MQTT in quanto la centralina verrà impostata in modo da leggere solo i comandi diretti a quella determinata auto.

La connessione con il broker è di tipo persistente, in modo tale che se la macchina perdesse la connessione, tutti i comadi inviati dal dispositivo mobile vengano recapitati alla ri-connessione della centralina

Ad esempio, mettiamo caso che io volessi inviare il comando di accensione delle 4 frecce alla macchina con IdMacchina = 13:

dispositivo mobile 🡪 publish su “car/13/cmd/4\_frecce”

*File JSON inviato*:

{

“Status”: true

}

Macchina 🡪 subscribe su “car/13/cmd/#”

In questo modo la nostra macchina sarà in “ascolto” su quel topic e riceverà tutti i comandi inviategli dal dispositivo mobile.

*Broker AMQP:*

Per la gestione dei dati inviati dalle macchine si è scelto di utilizzare un broker AMQP il quale riceverà un messaggio con queste informazioni:

RoutingKey: Milano

|  |  |
| --- | --- |
| Nome | Tipo |
| IdCar | Int |
| CarModel | String |
| BatteryLevel | Int |
| CurrentSpeed | Int |
| CurrentConsume | Float |
| CurrentPosition | String\* |
| BatteryStatus | Bool |
| DetectionDate | DateTime |

‘\*’: Il dato viene salvato in stringa poichè verrà formattato in questo modo “Lat: ... , Long: ...”.

Una volta ricevuto il dato, questo passa per l’exchanger che, a seconda della routingKey, indirizza il pacchetto verso una determinata coda.

Dato che a noi interessa salvare in un database separato solo ed esclusivamente i dati di Pordenone, sono state create due code:

* PordenoneQueue 🡪 coda dove finiranno tutte le rilevazioni delle macchine di Pordenone
* OtherCitiesQueue 🡪 coda dove finiranno tutte le altre rilevazioni

*HTTP server:*

Nel caso in cui nelle code popolate dall’exchanger ci fossero delle rilevazioni inerenti ai modelli Tesla, queste ultime vengono inviate tramite chiamate API ad un server. Il come vengano processati i dati non è richiesto dal problema, quindi possiamo ipotizzare che quando il “consumer” della coda trova una rilevazione di una Tesla, la invii tramite una chiamata POST al server HTTP.

*Esempio:*

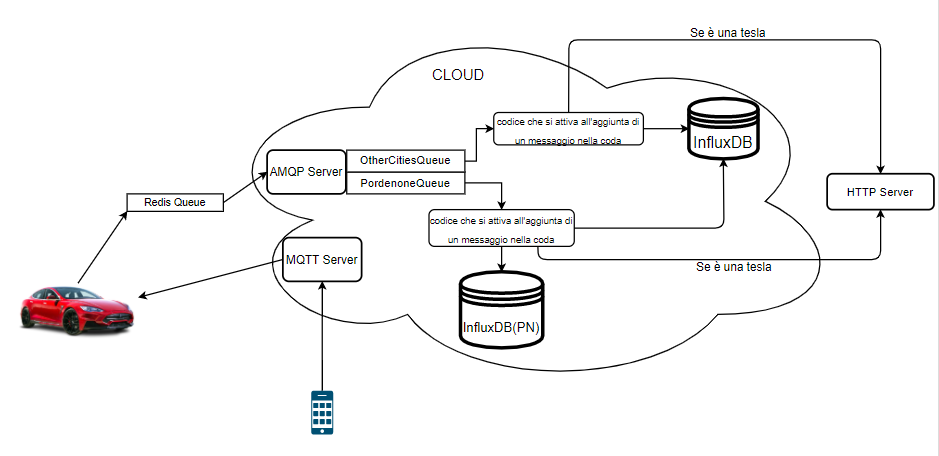
POST 🡪 https://www.carsIndustries.com/cars/*car\_id*/tesla/detection

*Database:*

Come database si è pensato di utilizzare InfluxDB in quanto offre anche la possibiltà di creare dei grafici statistici con i dati salvati al suo interno, inoltre essendo un DB non relazionale ci permette di fare le ricerche sulle rilevazioni in modo più veloce rispetto che ad un DB relazionale.

**Sviluppo**

Dopo un attenta analisi del problema, la soluzione proposta sarà:



**Conclusioni**

Nel problema è richiesto di revisionare il progetto nel caso in cui la centralina fosse meno prestante di quella iniziale, ma con questa soluzione andiamo a risolvere anche questo problema, in quanto se la coda non invia più dati, avremmo un riscontro sul DB, mentre per quanto riguarda la ricezioni dei comandi, utilizzando la connessione persistente la macchina riceverebbe i comandi una volta riconnessa al broker.